2nd Eurographics Workshop on

VIRTUAL ENVIRONMENTS

Workshop Sponsors:
Electricite de France,
Office of Naval Research, Europe
INA, IMAGINA
Fraunhofer IGD, Darmstadt
CEA, Villeneuve-St-Georges

Editor: Martin Göbel

Monte Carlo, Hotel Metropole,
4 Avenue Madone, 98000 Monaco

January 31st, - February 1st, 1995

Local organization :

Jacques David
CEA CEL-V/D.MA
94195 VILLENEUVE-St-GEORGES
FRANCE

Eurographics Workshop Proceedings Series ISSN 1024-0861

DRAFT SF 298

1. Report Date (dd-mm-yy) 31 Jan 95	2. Report Type workshop proceedings	3. Dates covered (from... to) 31 Jan - 1 Feb 1995
4. Title & subtitle 2nd Eurographics Workshop on Virtual Environments		5a. Contract or Grant # N00014--95-1-9003
		5b. Program Element #
6. Author(s) Martin Gobel, editors		5c. Project # 5202
		5d. Task # EUR
		5e. Work Unit #
7. Performing Organization Name & Address CEA CEL-V/D.MA 94195 Villeneuve-St-Georges France		8. Performing Organization Report # Eurographics Workshop Proceedings Series ISSN 1024-0861
9. Sponsoring/Monitoring Agency Name & Address Office of Naval Research Europe PSC 802 BOX 39 FPO AE 09499-0700		10. Monitor Acronym ONREUR
		11. Monitor Report #
12. Distribution/Availability Statement A		
13. Supplementary Notes		
14. Abstract		
15. Subject Terms		

Security Classification of			19. Limitation of Abstract	20. # of Pages	21. Responsible Person (Name and Telephone #)
16. Report Unclassified	17. Abstract Unclassified	18. This Page Unclassified	Unlimited		Mike Shear, 011-44-0171-514-4921

Second Eurographics Workshop on Virtual Environments

INTERNATIONAL PROGRAM COMMITTEE:

Frank Bagiana (ESA, The Netherlands)
Massimo Bergamasco (ARTS Lab, Italy)
Steve Bryson (Nasa Ames, US)
Jose Encarnacao (University Darmstadt, Germany)
Lennart Fahlen (SICS, S)
Wolfgang Felger (IGD, Germany)
Vartkes Goetcherian (CEC, B)
Roger Hubbold (Univ. Manchester, UK)
Robert Jacobson (WORLDESIGN,USA)
Hans Jense (TNO FEL, The Netherlands)
Heedong Ko (KIST, S.Korea)
Michael Moshell (Univ.o.Central Florida,USA)
Jens Neugebauer (IPA, Germany)
Philippe Queau (INA, France)
Lawrence Rosenblum (ONR, US)
Gurminder Singh (ISS, Singapore)
Jeffrey Shaw (ZKM, Germany)
William Sherman (NCSA, USA)
Mel Slater (Univ. of London, UK)
Robert Stone (ARRL, UK)
Daniel Thalmann (EPFL, CH)

Workshop Cochairpersons:

Martin Göbel
Fraunhofer Institut for Computer Graphics (IGD)
email: goebel@igd.fhg.de
Jaques David
CEA CEL-V/D.MA

Local Organizer:

Jaques David
CEA CEL-V/D.MA
email: david@limeil.cea.fr

WORKSHOP PROGRAM:

Tuesday, January 31st:

8.30-9.30: Registration

9.30- 9.45: Welcome

9.45-10.45: **Session 1: Experiments and Experience**

Is VR better than a Workstation ?
A Report on Human Performance Experiments in Progress
D. Mizell, S. Jones, P. Jackson ,D. Pickett (USA)

The influence of Dynamic Shadows on Presence in Immersive Environments
M. Slater, M. Usoh, Y. Chrysanthrou (UK)

10.45-11.00: Coffee Break

11.00-12.30: **Session 2: Systems and Platforms**

An Object Oriented Graphics System on the Way to VR
E. Beier (D)

Dynamic Rendering System - a fuzzy logic controlled System using
LoD technology optimized with Genetic Algortithms
R.D. Schraft, R.Dänighaus, J. Neugebauer(D)

VIPER (VIrtuality Programming Environment)-
A VR Application Design Platform
P. Torquet, R. Caubet (F)

12.30-14.00: Lunch Break

14.00-15.30: **Session 3: Interaction**

Fine Object Manipulation in Virtual Environments
R. Kijima, M. Hirose (J)

Virtual an Real Mulitsensor Grasping:
Interacting with Virtual Actors
Z. Huang, S. Rezzonico, R. Boulic, N. Magnenat Tahlmann,
D. Thalmann (CH)

Interaction Models for Speech Interfaces to VEs
J. Karlgren, I.Bretan, N. Frost, L. Jonsson (S)

15.30-15.45: Coffee Break

15.45-17.15: **Session 4: Metaphors and Legibility**

Modern Holy Grail: The Metaphor Towards a Methodology of Choice
O. Dupuy (F)

Improving the Legibility of Virtual Environments
R. Ingram, S. Benford (UK)

20.00 Workshop Dinner

Wednesday, February 1st:

9.00-10.30: **Session 5: Algorithms**

Fast Algorithms for drawing Nonuniform B-sline Surfaces:
a practical Application in Virtual Environments
M. Bergamasco, D. Bucciarelli, P. Degl'Innocenti (I)

Evolutionary Art in Virtual Worlds
I. Soo Lim (SG)

Generating Multi Levels of Detail from Polygonal Geometry Models
G. Schaufler, W. Stürzlinger (A)

10.30-10.45 Coffee Break

10.45-11.45: **Session 6: Facial Expression**

Searching for Facial Expressions by Genetic Algorithms
J.-H. Kim, H.-D. Ko, J.-H.Kim (KR)

Coordinating Vocal and Visual Parameters for 3D Virtual Agents
C. Pelachaud (I),S. Prevost (US)

11.45-12.45: **Session 7: Application**

Virtual Urban Environment for the Simulation of an Automated Electrical Cars
Platon in the Praxitele Project
R. Cozot, S. Donikian (F)

An Interactive Virtual Experience - The SBG Roadshow
P. Astheimer, W. Felger (D)

12.45-14.00: Lunch Break

14.00.15.30: **Session 8: Scientific Visualization Application**

Virtual Environment Techniques for the Exploration of Earth Observation Data
F. Bagiana, H. Jense (NL)

Virtual Reality Techniques for Scientific Visualization
J.L. Pajon, V. Bui Tran, P. Vuyslteker, P. Guilloteau, J. David (F)

Further Development of the Responsive Workbench
B. Fröhlich, B. Kirsch, W. Krüger, G. Wesche (D)

15.30-15.45 Coffee Break

15.45-16.15: Panel Discussion on Realism vs. Realtime

16.15-16.30: Conclusions and Closing

17.00: Program committee meeting

Is VR Better than a Workstation?
A Report on Human Performance Experiments in Progress

David Mizell
Stephen Jones
Research & Technology Organization
Boeing Computer Services

Paul Jackson
Texas A&M University

Dasal Pickett
Morris Brown College

Extended Abstract
October 14, 1994

1. Introduction

A scientist working on virtual reality in an industrial research environment frequently gets questioned in the following vein by ruthless men in $1000 suits: "Why do I need VR? What can I do in VR that I can't do just as well on a workstation with fast graphics? For what application is VR so much more capable that it justifies spending ten times as much for the hardware?"

For most of the brief history of this technology, the answer to such questions was unpleasantly easy. Because of the primitive quality of VR systems, there was no application, other than entertainment, where a VR system could be expected to pay for itself. This is probably the primary cause of today's rather striking dearth of non-entertainment VR applications in routine use. For several years, the equipment simply wasn't good enough. The displays were abysmally low-resolution, the gloves too inaccurate, the head trackers too slow or noise-sensitive, and the rendering performance was inadequate for any but cartoon-like virtual worlds. Accordingly, for most of the time that most of us have been working in this field, there was no point in trying to experimentally compare VR systems to graphics workstations -- whatever its intrinsic merits might be, the VR system would have lost every time.

The vendors have steadily been improving the quality of VR hardware and software, however. We can now get color HMDs with 640x480 resolution, there is a new generation of faster head trackers, we can obtain ever-faster graphics rendering hardware, and so on. By now, it is reasonable to consider comparing VR to other visualization technologies. The experiments reported here were inspired by this perception about improving VR hardware and by the experimental work done by Randy Pausch's group at UVa in comparing head-coupled to non-head-coupled searching of a virtual space [Pausch et al Frontiers '93].

2. The Experimental Hypothesis

We had a general, imprecise hypothesis we wanted to experimentally explore and quantify to the extent possible: that VR helps a user comprehend -- build a mental model of -- complex 3D geometry. If an aircraft hydraulics engineer has to design a single 2-ft. section of hydraulic tubing, then a CAD workstation is perfectly adequate. She/he can enter and modify the tube design, view it from any angle, and so on. If, however, the designer wants to view this tube in its installed location among the complex tangle of hydraulics and other systems inside the main landing gear wheel well, the view on the workstation screen intuitively does not seem to be as useful as a view of the same geometry within a virtual environment.

Perhaps the ultimate payoff for VR is in the potential for interaction. Our hydraulics engineer could conceivably move tubes around and change their shape, all in first person, accomplishing a

design task with much more speed and understanding of the consequences of each design decision than on a workstation. We hypothesized, however, that even if we limited ourselves to visualization, in the absence of any sort of interaction capability, that VR might outperform a workstation. The basic notion is that, because of the way the brain correlates information from the visual system and the proprioceptive sense, plus motion parallax, plus the advantage of stereo vision, that VR might give a user a more accurate understanding of complex assemblages of 3-D geometry than a workstation view.

We had the following sort of anecdotal evidence: a Boeing engineer assigned to the Space Station project, who had been thoroughly involved in the CAD design of space station components, when viewing the CAD geometry of the Lab Module in our VR system for the first time, exclaimed "Wow! It's so big!" He had looked at the same geometry on his workstation for more than a year, but it had always looked small on his screen. Then he was suddenly placed in a position that appeared to him as if he were a few feet from the entrance to the lab module, and he discovered that he bend his neck from very high to very low to look from the top of the design to the bottom.

3. The Experiment Design

We debated many different approaches to experiments that might shed light on this hypothesis. Some placed the user inside a virtual space with virtual objects possibly on all sides around the user. Others showed the user a set of objects of relatively small size, all viewable by the user at once. We settled on an experiment of the latter type, not because we believed that this class of application were likely to yield the highest leverage for VR (most of us believed the opposite), but because we considered it the simplest experiment for us to get started with.

We created some three-dimensional puzzles. The base of each puzzle was a sheet of pegboard, 18 inches by 16 inches. We obtained some aluminum rods of .25 in. diameter, which fitted snugly into the holes in the pegboard. We bent the rods into a variety of shapes and painted them different colors. Selecting from four to eight of the rods, we would insert them into a pegboard, creating a sort of three-dimensional "wire sculpture." [We'll include pictures of these in the paper.] We bent and painted pairs of the rods identically, so that we could construct two instances of each puzzle we designed. Using a CAD package, we also created 3-D solid models of each of the puzzles we designed.

The task each of our volunteer experimental subjects were given was to build a duplicate copy of a puzzle. He or she would be given an empty pegboard base and a randomly-arranged set of bent rods. This set would include all the rods needed to build a replica of the puzzle in question and a few extra ones. The key variable in the experiment was the way the subject would be shown what the completed sculpture, which they were to duplicate, looked like. They would either be shown (1) an actual physical pegboard with the rods inserted in the correct positions, (2) a CAD model of the completed puzzle on a workstation screen, or (3) a CAD model of the completed puzzle, viewed in a stereo VR display: a Fakespace boom. In each case, the representation of the solved puzzle was placed close to the table where the subject was constructing the duplicate, so he/she could refer to to solution as often has he/she wished, from whatever point of view he/she wished. In order to minimize the influence of user interface issues, the workstation was equipped with a Polhemus tracker mounted on a small wooden plaque. If the subject moved or rotated the wooden plaque, the view of the completed puzzle on the workstation screen would move or rotate correspondingly. Using the completed physical puzzle as a reference for constructing a duplicate was our experimental control. We wanted to compare the performance difference between the workstation and the VR references.

We (subjectively) selected three levels of difficulty among the puzzles. Each subject would go through a coached training session with a very simple puzzle, and then undergo three timed and scored experiments. We randomly

permuted the three difficulty levels and the three methods of displaying the solution with each subject. We devised a scoring system for grading the accuracy of the replication of the puzzle. It was based on whether the subject used the correct rod, whether the rod was in the correct hole in the pegboard, and how near the rod was to the correct rotational orientation.
(The paper will go into more depth on the statistical methods used and provide a tabular summary of the results.)

4. The Results

Very briefly, here is the outcome of this experiment: people had faster times using the workstation, but they achieved higher accuracy scores using the boom (the physical representation was, of course, superior to the others in both respects). It isn't hard to guess why the boom had slower times. With the physical model or with the workstation, a subject could quickly glance back and forth between the completed puzzle and his/her duplicate. There was no quick glancing with the boom. The subject had to move a greater distance, grasp the boom handles and place his/her face against the display. This made us wish we had a "hybrid" HMD that could switch under software control between occlusive and transparent. That way, when a subject looked toward the virtual representation of the completed puzzle, the display would show the graphical object, but when the subject looked back at his/her physical duplicate, the display would turn transparent. Such a display would make glancing back and forth between the solution and the duplicate as easy as in the physical case.

5. Future Work

This is definitely work in progress. There are many more experiments of this sort we could do. In the future, we want to look at experiments using models that are "bigger" than the subject, i.e., virtual worlds that surround the user. Intuition says that the workstation would be at a considerable disadvantage in that case.

There are many open issues with respect to experimentally comparing VR with other human-computer interfaces:

- are we testing the right hypotheses, with the right experiments?
- how do we factor out accidental influences, such as unfamiliar or intimidating user interfaces? Some subjects were very uncomfortable using the boom, for example.

Of these, the most critical is the design of meaningful experiments. There is room for many more players in this game.

The Influence of Dynamic Shadows on Presence in Immersive Virtual Environments

Mel Slater, Martin Usoh, Yiorgos Chrysanthou,
Department of Computer Science, and
London Parallel Applications Centre,
QMW University of London,
Mile End Road,
London E1 4NS, UK.

1. Introduction

We describe an experiment to examine the effect of shadows on two different aspects of the experience of immersion in a virtual environment (VE): depth perception and presence. It is well-known that shadows can significantly enhance depth perception in everyday reality [1,5,7]. Shadows provide alternative views of objects, and provide direct information about their spatial relationships with surrounding surfaces. VR systems typically do not support shadows, and yet potential applications, especially in the training sphere, will require participants to make judgements about such relationships. Even the simple task of moving to an object and picking it up can be problematic when observers cannot easily determine their distance from the object, or its distance from surrounding objects. We introduce dynamic shadows to examine whether such task performance can be enhanced.

We have argued elsewhere [8] that presence is the key to the science of immersive virtual environments (virtual reality). We distinguish, however, between immersion and presence. Immersion includes the extent to which the computer displays are extensive, surrounding, inclusive, vivid and matching. The displays are more extensive the more sensory systems that they accommodate. They are surrounding to the extent that information can arrive at the person's sense organs from any (virtual) direction. They are inclusive to the extent that all external sensory data (from physical reality) is shut out. Their vividness is a function of the variety and richness of the sensory information they can generate [11]. In the context of visual displays, for example, colour displays are more vivid than monochrome, and displays depicting shadows are more vivid than those that do not. Vividness is concerned with the richness, information content, resolution and quality of the displays. Finally, immersion requires that there is match between the participant's proprioceptive feedback about body movements, and the information generated on the displays. A turn of the head should result in a corresponding change to the visual display, and, for example, to the auditory displays so that sound direction is invariant to the orientation of the head. Matching requires body tracking, at least head tracking, but generally the greater the degree of body mapping, the greater the extent to which the movements of the body can be accurately reproduced.

Immersion also requires a self-representation in the VE - a Virtual Body (VB). The VB is both part of the perceived environment, and represents the being that is doing the perceiving. Perception in the VE is centred on the position in virtual space of the VB - e.g., visual perception from the viewpoint of the eyes in the head of the VB.

Immersion is an objective description of what any particular system does provide. Presence is a state of consciousness, the (psychological) sense of being in the virtual environment. Participants who are highly present should experience the VE as more the engaging reality than the surrounding world, and consider the environment specified by the displays as places visited rather than as images seen. Behaviours in the VE should be consistent with behaviours that would have occurred in everyday reality in similar circumstances.

Presence requires that the participant identify with the VB - that its movements are his/her movements, and that the VB comes to "be" the body of that person in the VE. We speculate that the additional information provided by shadows about the movements of the VB in

relationship to the surfaces of the VE can enhance this degree of association, and hence the degree of presence. However, we were unable to test this in the current experiment. We do, however, consider the proposition that shadows, increasing the degree of vividness of the visual displays, will enhance the sense of presence.

2. Experiment

2.1 Scenario

The experimental scenario consisted of a virtual room, the elevation of which is shown in Figure 1. Five red spears are near a wall, but behind a small screen. Another green spear is at position G. The subject begins the experiment by moving to the red square (X), and facing the spears. The instruction is to choose the spear nearest the wall, observing from position X. Having chosen that spear, the subject moves towards it, picks it up and returns to X. There the subject turns to the left, facing a target on the far wall. The subject must orient the spear to point approximately towards the target, fire and guide it towards the target by hand movements. The instructions were that the spear must be shot at the target, and that it must be stopped the instant that its point hit the target. Finally, the subject must bring the green spear to position X. This was repeated six times for each subject.

Prior to the start of the experiment each subject was given a sheet explaining these procedures, and the first run was for practice, the experimenter talking the subject through the entire scenario. Runs 1 through 5 were carried out by the subject without intervention by the experimenter. Between each run the subject was advised to relax with closed eyes, either with or without the head-mounted display (HMD, see below), although all but one continued to wear it during the two minutes that it took to load the program for the subsequent run. Each of the five runs were the same apart from the distances of the red spears from the wall. Also, some runs displayed dynamic shadows of the spears and the small screen, while others did not.

Eight subjects were selected by the experimenters asking people throughout the QMW campus (in canteens, bars, laboratories, offices) whether they wished to take part in a study of "virtual reality". People from our own Department were not included.

The design is shown in Table 1, which indicates the positions of the point-light source for those runs that included shadows. Note that of the 40 runs, 20 included shadows.

Table 1

Runs of the Experiment for Each Subject

1,2,3,4 denotes the four point-light positions of Figure 1

0 denotes no shadows

Subject	No. shadow scenes / 5	Run 1	Run 2	Run 3	Run 4	Run 5
1	1	0	0	2	0	0
2	1	0	0	2	0	0
3	2	0	0	2	3	0
4	2	0	0	2	3	0
5	3	0	1	2	3	0
6	3	0	1	2	3	0
7	4	0	1	2	3	4
8	4	0	1	2	3	4

2.2 Spatial Variables and Hypotheses

The variables measured in order to assess the effects of shadows on spatial judgement were as follows:

Spear Selected
S: the spear selected from observation position X. The spears ranged from 50 cm to 90 cm from the wall, positioned with 10 cm variations. The small screen in front of the spears obscured the positions where they touched the floor, for any subject standing at position X. Also, because their distances from the wall varied only slightly, their heights, as judged from position X would look the same. It was therefore very difficult to judge which spear was nearest the wall. Variable S was the rank order of the spear chosen, where 1 would be the nearest to the wall, and 5 the furthest.

The hypothesis was that subjects would be able to use the shadows of spears on the walls to aid their judgement about the closeness to the walls, so that those runs that included shadows would result in a greater number of correct spears being chosen.

Distances from Target
C: this is the distance of the point of the spear from the centre of the target at the position that it was stopped in flight by the subject.

The hypothesis was that the subjects would be able to use the shadow of the spear in flight, especially its shadow on the target wall, to help guide the spear towards the target. Therefore, the mean distance should be less for the shadow runs than for the non-shadow runs.

D: this is the distance that the point of the spear was behind or in front of the target at the position that it was stopped by the subject.

The hypothesis is as for C, except that here we would expect a greater shadow effect since the action required to stop the spear in flight (releasing a button on the hand-held 3D mouse) is simpler than that involved in guiding the spear to the bulls eye. Moreover, at the moment the spear point touched the target wall, it would also meet its shadow.

2.3 Presence Variables and Hypotheses

In previous studies we have used subjective reported levels of "presence" based on a questionnaire. In this method subjective presence was assessed in three ways: the sense of "being there" in the VE, the extent to which there were times that the virtual world seemed more the presenting reality than the real world, and the sense of visiting somewhere rather than just seeing images. In the present study these three basic determinants were elaborated into six questions, each measured on a 7-point scale, where lowest presence is 1, and highest is 7 (see Appendix A). The overall presence score (P) was conservatively taken as the number of high (6 or 7) ratings amongst the six questions, so that $0 \leq P \leq 6$.

Although we have obtained good results with such subjective measures before, in the shadow experiment we introduced in addition a more "objective" measurement of presence. This was achieved by having one particular object (a radio) in both the real world of the laboratory in which the experiment took place and the virtual world of the room with spears.

Just before the practice run the subjects were shown a radio on the floor against a large screen in the laboratory. They were told that they would see "the radio" in the virtual world, and that occasionally it would switch itself on. Whenever they heard the sound they should point towards "the radio", and press a button on the hand-held mouse. This would act as an "infra-red" device to switch the radio off. Before they entered into the VE the radio was momentarily switched on, deliberately not tuned to any particular channel therefore causing it to play an audible but meaningless tone. Each time that the subject entered into the VE, i.e., at the start of

each run they were told: "Orient yourself by looking for the red square on the floor and the radio". The radio was placed in the VE at the same position relative to the red square as the real radio was to the position of the subject just before entering the VE.

At four moments during the experiment, always while the subject was (virtually) on the red square, the real radio was moved to one of four different positions. These were 1m apart from each other, on a line coincident (in the real world) with the small screen by which the radio was located (in the virtual world). The ordering was selected randomly before the start of the experiment. The virtual radio was always in the same place. Therefore the subject would hear the sound coming from a different location compared to the visible position of the radio. The idea is that (other things being equal), a high degree of presence would lead to the subject pointing towards the virtual radio rather than the real one. Hence we tried to cause and use the conflict between virtual and real information as an assessment of presence. Those (two) subjects who did ask about the contradiction were told "Just point at where you think the radio is". Throughout, both the real radio and the virtual radio were referred to as "the radio", deliberately allowing for a confusion in the minds of the subjects.

It is important to note that we mean "presence" in a strong behavioural sense with respect to this measurement. The questionnaire attempts to elicit the subject's state of mind. The radio method though is concerned only with their behaviour. If they pointed to the virtual radio because of a need to obey the experimenter, or because it was a matter of "playing the game", then so be it. Provided that they act in accordance with the conditions of the VE, this is behavioural presence.

Let R be the angle between the subject's real pointing direction and the direction to the real radio. Let V be the angle between the subject's virtual pointing direction and the direction to the virtual radio. Small V therefore occurs when the subject points towards the virtual radio. We use $P_a = R/V$ as the measurement of the extent to which the subject tends towards the virtual radio - a small V in comparison to R would result in large P_a. Therefore larger values of P_a indicate greater tendency towards the virtual.

There were two hypotheses relating to P_a: First, that it would correlate positively with P, and second that the greater exposure of the subject to shadows, the greater the value of P_a. Of course, we would also expect that the greater the exposure to shadows, the greater the value of P.

2.4 Representation System Dominance

A clear objection to this procedure is that it could be measuring the extent of visual or auditory dominance rather than presence. Faced with conflicting information from two senses, the resulting action is likely to depend on which sensory system is "dominant". In previous work [9,10] we have explored the relationship between dominant representation systems and the extent of subjective presence, and have always found a very strong relationship. This is based on the idea that people differ in the extent to which they require visual, auditory or kinesthetic/tactile information in order to construct their world models, and that each person may have a general tendency to prefer one type of representation (say visual) over another (say auditory). We found that in experiments where the virtual reality system presented almost exclusively visual information, the greater the degree of visual dominance the higher the sense of presence, whereas the greater degree of auditory dominance, the lower the sense of presence.

In this shadow experiment therefore we employed an updated version of the questionnaire we used in [10] that is given to the subjects before attending the experimental session, that attempts to elicit their preferences regarding visual, auditory and kinesthetic modes of thinking. This questionnaire presents 10 situations, each one having three responses (one visual, one auditory, and one kinesthetic response). They are asked to rank their most likely response as 1, next most likely as 2, and least likely as 3. From this a V score is constructed as the total

number of V=1 scores out of 10, and similarly for A and K. Alternatively the sums of the responses may be used. These V and A variables can therefore be used to statistically factor out the possible influence of visual or auditory dominance on the radio angles.

The hypothesis with respect to V, A and K would be that V and K would be positively correlated with presence (however it is measured) whereas A would be negatively correlated, in line with our previous findings. Note that by construction, there are only 2 degrees of freedom amongst V, A and K.

3. Apparatus

3.1 Equipment

The experiments described in this paper were implemented on a DIVISION ProVision system, a parallel architecture for implementing virtual environments running under the dVS (v0.1) operating environment. The ProVision system is based on a distributed memory architecture in which a number of autonomous processing modules are dedicated to a part of the virtual environment simulation. These processing modules or Transputer Modules (TRAMs) are small self-contained parallel processing building blocks complete with their own local memory and contain at least one Inmos Transputer which may control other specialised peripheral hardware such as digital to analog converters (DAC). Several modules exist. These include:

- the module to act as the module manager.
- the DAC module for audio output.
- polygon modules for z-buffering and Gouraud shading.
- application specific modules for the user applications.

The dVS operating environment (Grimsdale, 1991) is based on distributed Client/Server principles. Each TRAM or processing cluster is controlled by an independent parallel process known as an Actor. Each provides a set of services relating to the elements of the environment which it oversees. Such elements presently consist of lights, objects, cameras, controls (i.e. input devices), and collisions between objects. Thus, an Actor provides a service such as scene rendering (visualisation actor). Another Actor may be responsible for determining when objects have collided (collision actor) and yet another for hand tracking and input device scanning. All these Actors are co-ordinated by a special Actor called the Director. Communication between the different Actors can only be made via the Director. The Director also ensures consistency in the environment by maintaining elements of the environment which are shared by the different Actors.

The ProVision system includes a DIVISION 3D mouse, and a Virtual Research Flight Helmet as the head mounted display (HMD). Polhemus sensors are used for position tracking of the head and the mouse. The displays are colour LCDs with a 360×240 resolution and the HMD provides a horizontal field of view of about 75 degrees.

All subjects saw a VB as self representation. They would see a representation of their right hand, and their thumb and first finger activation of the 3D pointer buttons would be reflected in movements of their corresponding virtual finger and thumb. The hand was attached to an arm, that could be bent and twisted in response to similar movements of the real arm and wrist. The arm was connected to an entire but simple block-like body representation, complete with legs and left arm. Forward movement was accompanied by walking motions of the virtual legs. If the subjects turned their real head around by more than 60 degrees, then the virtual body would be reoriented accordingly. So for example, if they turned their real body around and then looked down at their virtual feet, their orientation would line up with their real body. However, turning only the head around by more than 60 degrees and looking down (an infrequent occurrence), would result in the real body being out of alignment with the virtual body.

The 3D mouse is shaped something like a gun. There is a button in the position of the hammer, which is depressed by the thumb. This causes forward motion in the direction of pointing.

There is a button on each side of this central thumb button, each activated by the thumb. The left one was used to fire the spears - while this button was depressed the spear would move in a direction determined by hand orientation. The spear would stop on release of this button, and could not be activated again, thus giving the subject one chance per spear. The right thumb button was used as the "infra-red" radio switch. Corresponding to the trigger is a button for the forefinger. This is used to pick objects - squeezing this finger button while the virtual hand intersects an object results in the object attaching to the hand. Subjects were able to master these controls very quickly.

3.2 Shadow Algorithm and Frame Rates

The shadow algorithm is described in detail elsewhere [3]. It is based on a dynamic Shadow Volume BSP tree [2], constructed from polygons in arbitrary order, that is without the necessity of a separate scene BSP tree. Shadows are created as polygons in object space. Creation of new shadows and changes to shadows are communicated dynamically to the renderer via the Director.

For reasons described below, the entire scene was small, consisting of 413 triangles, of which only 52 would be likely to influence shadow creation. The frame rate achieved without shadows was 9Hz. The frame rate with shadows, 6 to 8Hz, was not very satisfactory, but due to the particular version of the dVS software architecture in use on this machine at the time of the experiment.

Without rendering the shadow algorithm runs on this machine at a frequency of between 19 and 21Hz depending on the complexity of the view at any moment. The renderer does not however run at this frequency during dynamic changes of a virtual object, due to update problems associated with the extant implementation of the dVS dynamic geometry object. Therefore, when rendering and the associated communication time is included, the frame rate is 6 to 8Hz. (A new version of dVS is intended to solve this problem).

dVS v0.1 maintains the concept of a "dynamic geometry object". This is a vertex-face structure representing a (possibly empty) set of polygons. The actual polygons belonging to this object can be created or modified at run time. When such a change is made to a dynamic object, there is an "update" generated that sends the object to the Director for distribution to the Visualisation Actor and then onto to the renderer.

Upon any change of a virtual object the shadow algorithm recomputes the shadow scene outputting any modified shadow polygons, i.e. any polygons that have been deleted and any that have been created. This information is transmitted to the shadow generation module which will mark deleted polygons as invisible to be re-used later by new shadow polygons. The module uses a linked list structure of dynamic objects - the shadow object. Each element in the list is a dynamic object consisting of 32 shadow polygons. This linked list structure is necessary in order to break down the entire list of potential shadow polygons into smaller chunks, rather than have one dynamic geometry object for all possible shadows, since the dynamic geometry implementation can only send updates of an entire dynamic object to the Visualisation Actor. Note that a change in one single shadow polygon will result in the communication of a complete 32-polygon dynamic object. If, unfortunately, 33 shadow polygons change, then two dynamic objects consisting of 64 polygons are communicated, and so on.

Note that there is one important implication of this for the spatial judgement component of the experiment - obviously the spear travels more slowly when there are shadows. Without shadows the mean velocity is 92 cm/sec, and with shadows 47 cm/sec. Therefore it can validly be argued that differences in targeting performance result from the velocity rather than the use of shadows. However, the effect of this can be examined statistically. With regard to the influence on presence we would argue that the slower frame rate in the case of shadows would tend to have a negative effect on presence.

4. Results

4.1 Spatial Variables

Spear Selected

Shadows made no difference at all to the selection of the "correct" spear (the one closest to the wall).

Distances from Target

Consider first C the distance of the point of the spear from the centre of the target. A regression analysis was used to examine the effect of velocity, showing that velocity within each of the shadow/ no-shadow groups was did not have a statistically significant effect. The mean distance without shadows is 152cm and 115cm with shadows. However, the difference between these two is not statistically significant.

Consider next D, the perpendicular distance of the point of the spear from the wall of the target. This could be positive (spear stops in front of the target) or negative, the spear stops behind). Carrying out a within-group regression analysis to examine the effect of velocity again shows that velocity is not statistically significant. The means are -39.9cm without shadows, and 3.3cm with shadows. The standard errors are 3.6 and 3.5 respectively and the difference is significant at 5%. The medians of the shadow and non-shadow D values are -3cm and -38cm respectively.

Although the within-group velocity appeared not to be statistically significant in each case, there is still some doubt about whether the inference about better performance in the case of shadows is safe. The variation of velocity within groups was not very great (the minimum and maximum velocities were 81.6 to 99.0 for the non-shadow group, and 36.0 to 60.4 for the shadow group). Subsequent experiments should attempt to produce a greater similarity in performance between the two groups.

4.2 Presence

Subjective Presence

P is the number of "high" questionnaire scores, as a count out of 6. We therefore treated P as a binomially distributed dependent variable, and used logistic regression.

In logistic regression [4], the dependent variable is binomially distributed, with expected value related by the logistic function to a linear predictor. Let the independent and explanatory variables be denoted by $x_1, x_2, \ldots, x_k$. Then the linear predictor is an expression of the form:

$$\eta_i = \beta_0 + \sum_{j=1}^{k} \beta_j x_{ij} \quad (i = 1,2,\ldots,N) \qquad (1)$$

where N (=8) is the number of observations. The logistic regression model links the expected value $E(P_i)$ to the linear predictor as:

$$E(P_i) = \frac{n}{1+\exp(-\eta_i)} \qquad (2)$$

where n (=6) is the number of binomial trials per observation.

Maximum likelihood estimation is used to obtain estimates of the β coefficients. The deviance (minus twice the log-likelihood ratio of two models) may be used as a goodness of fit significance test, comparing the null model ($\beta_j = 0$, j = 1,...k) with any given model. The change in deviance for adding or deleting groups of variables may also be used to test for their significance. The (change in) deviance has an approximate χ^2 distribution with degrees of freedom dependent on the number of parameters (added or deleted).

Table 2

Logistic Regression Equations

$\hat{\eta}$ = fitted values for the presence scale

A = Auditory Sum, NS = number of shadows

Standard Errors shown in brackets

Model
$\hat{\eta}$ = 15.0 + 0.7*NS - 9.5*A (3.7) (0.4)

Overall Deviance =3.454, d.f. = 5

χ^2 at 5% on 10 d.f. = 11.070

Deletion of Model Term	Change in Deviance	Change in d.f.	χ^2 at 5% level
NS	4.123	1	3.841
A	9.088	1	3.841

Table 3

Normal Regression Equations

$\hat{P}_a$ = fitted values for the angular discrepancy

NS = number of shadows

Group	Model
Visually dominant	$\hat{P}_a$ = -13.6 + 10.6*NS (3.7)
Auditory dominant	$\hat{P}_a$ = 9.427 + 0.08*NS (3.7)

Multiple Correlation Coefficient, R^2 =0.29, d.f. = 36

Table 2 shows the result of the fit with P as the dependent variable, and the number of shadow runs (NS) and the auditory sum score (A) as the explanatory variables, across the 8 subjects. These were the only statistically significant variables found, and this supports the hypothesis that subjective presence is positively related with the shadow effect. As we have found previously, given this exclusively visual VE, the greater auditory dominance, as measured by the sum of A responses to the pre-questionnaire, the less the reported subjective presence.

Angular Discrepancy

Here we take P_a as the dependent variable and carry out a Normal regression with number of shadows (NS) and the representation system scores as the explanatory variables. NS proved

once again to be significant and positively related to P_a. However, the V, A and K variables were not significant. Nevertheless it seemed important to try to rule out the possibility that the result with the angular discrepancy was simply due to visual or auditory dominance. Therefore a new factor was constructed, "sensory dominance" which has the value 1 if V>A otherwise 2. Hence this directly refers to visual or auditory dominance. The result of the regression analysis including this was interesting: for those who were visually dominant, there is a significant positive relationship between P_a and NS, whereas there is no significant relationship for those who were dominant on the auditory score. This is shown in Table 3. (It so happened that 4 of the subjects were visually dominant).

5. Conclusions

There are three main issues : First, the point of this paper is not that we have an algorithm that can generate shadow umbrae rapidly in dynamically changing scenes. Even in this very small scene the rendering frame rate was no where near adequate. There is clearly a lot of work to do in the location of this algorithm in the system architecture, in order to obtain maximum performance by minimising communication bottlenecks.

Second, although we have considered depth and spatial perception problems in the experiment, again, this is not the major point. It is more or less obvious, from everyday reality, and from perceptual studies that shadows do indeed enhance depth perception, and that we are better off with them than without them. Moreover, our experimental design in this regard was not ideal, since we did not control a factor (velocity) that potentially has an impact on the results.

Third, the real point of the experiment was the examination of the relationship between dynamic shadows and the sense of presence. This result is not obvious, and was motivated by the idea that presence is (amongst other things) a function of immersion, and immersion requires vividness. We used two independent measures - one subjective from the post-experiment questionnaire, and the other objective, as a ratio of angles of real to virtual pointing directions. Each method gave similar results, and the two measures were significantly correlated. Moreover, we found that for those people who were more visually dominant their (angular ratio) presence increased with exposure to shadows but that this did not hold for those who were dominant on the auditory scale. Increase in the subjective presence scale was also associated with an increase in shadow exposure, but with a decrease in the auditory scale. These results also support our earlier findings regarding the importance of the sensory system preferences in explaining presence.

We suspect that much stronger results on presence would have been obtained had we been able to allow the virtual body to cast shadows. However, this was not practical given the communication bottleneck problems discussed in §3.2.

If an application does not require presence, there is little point in using a virtual reality system. If a virtual reality system is used for an application, then there is little point to this unless it can be shown that a sense of presence is induced for most of the potential participants. If the results of our shadow experiment are confirmed by later studies then it will have been shown that the great computational expense of shadow generation is worth-while for those applications where the participants are likely to be "visually dominant".

Acknowledgements
This work is funded by the U.K. Engineering and Physical Sciences Research Council (EPSRC), the Department of Trade and Industry, and DIVISION Ltd, through grant CTA/2 of the London Parallel Applications Centre. Thanks to David Sweeting of Aeronautical Engineering Department of QMW, for helping to find subjects for the experiment.

References

[1] Cavanagh, P., Leclerc, Yvan G. (1989) Shape from shadows. Journal of Experimental Psychology: Human Perception & Performance, 15 (1), 3-27.

[2] Chin, N., Feiner, S. (1989) Near Real-Time Shadow Generation Using BSP Trees, Computer Graphics 23(3), 99-106.

[3] Chrysanthou, Y., Slater, M. (1995) Shadow Volume BSP Trees for Computation of Shadows in Dynamic Scene, ACM SIGGRAPH Symposium on Interactive 3D Graphics (April, 1995).

[4] Cox, D.R. [1970] *Analysis of Binary Data*, London: Menthuen.

[5] Gregory, R.L. (1990) Eye and Brain: The Psychology of Seeing, Fourth Edition, Weidenfield and Nicholson, 182-187.

[6] Grimsdale, C. (1991). dVS - Distributed Virtual environment System, Proceedings of Computer Graphics 91 Conference, London.

[7] Puerta, A.M. (1989) The power of shadows: shadow stereopsis. Journal of the Optical Society of America, 6, 309 - 311.

[8] Slater, M. A. Steed and M. Usoh (1994) Steps and Ladders in Virtual Reality, ACM Virtual Reality Science and Technology (VRST), eds G. Singh and D. Thalmann, World Scientific, 45-54.

[9] Slater, M. and M. Usoh, Representation Systems, Perceptual Position and Presence in Virtual Environments, Presence: Teleoperators and Virtual Environments, 2.3 MIT Press, 221-234.

[10] Slater, M., M. Usoh, A. Steed, Depth of Presence in Virtual Environments, Presence: Teleoperators and Virtual Environments, 3.2 (in press).

[11] Steuer, J. (1992) Defining Virtual Reality: Dimensions Determining Telepresence, Journal of Communication 42(4), 73-93.

Appendix A: Presence Questions

All questions were answered on a 1 to 7 scale, not reproduced here for space reasons.

1. Please rate *your sense of being there* in the virtual reality.

2. To what extent were there times during the experience when the virtual reality became the "reality" for you, and you almost forgot about the "real world" of the laboratory in which the whole experience was really taking place?

3. When you think back about your experience, do you think of the virtual reality more as *images that you saw*, or more as *somewhere that you visited* ?

4. During the course of the experience, which was strongest on the whole, your sense of being in the virtual reality, or of being in the real world of the laboratory?

5. When you think about the virtual reality, to what extent is the way that you are thinking about this similar to the way that you are thinking about the various places that you've been today?

6. During the course of the virtual reality experience, did you often think to yourself that you were actually just standing in a laboratory wearing a helmet, or did the virtual reality overwhelm you?

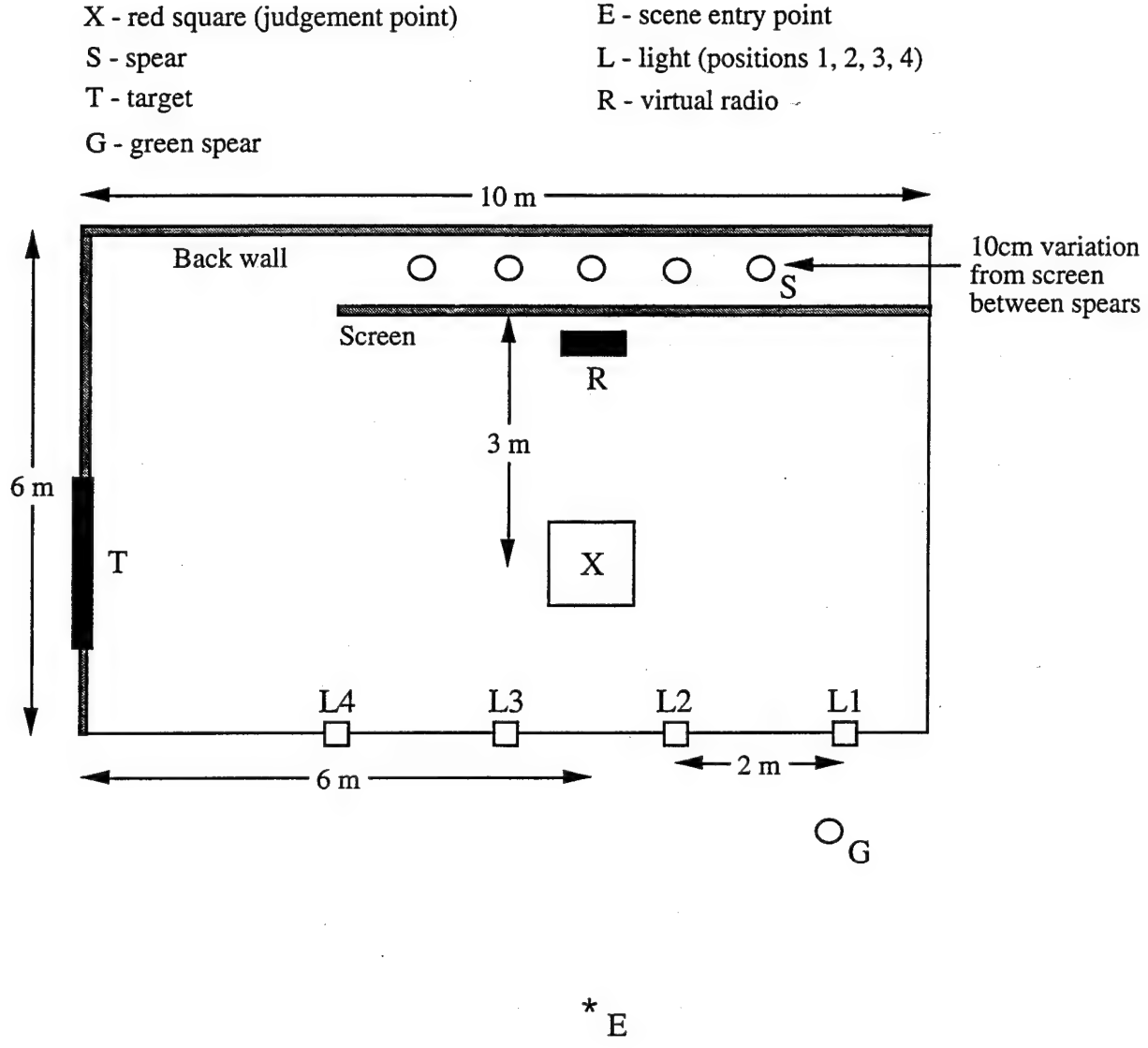


Figure 1
Plan View of the Virtual Environment

An Object-Oriented Graphics System on the Way to Virtual Reality

Ekkehard Beier
TU Ilmenau, Computer Science
Department of Computer Graphics
email: *ekki@prakinf.tu-ilmenau.de*
yart/vr: *institute@speedy.prakinf.tu-ilmenau.de*

Abstract

The paper describes some innovative concepts in the design of an object-oriented graphics kernel that could be useful for future applications in a network-transparent virtual reality environment. First, the benefits of object-orientation in graphics will be briefly explained. Afterwards, some concepts of the YART graphics kernel [Bei94a, Bei94d] will be described, which allow the use of YART in a heterogeneous, network-distributed virtual reality environment. The main features are graphical output, interaction and interpretative language binding. The presentation of an initial VR package on top of YART and some views on future research fields conclude the paper.

1 Object-Orientation in Graphics

This section briefly describes the benefits of object-orientation in 3D graphics and gives an overview of the current state of the art.

1.1 Conformity with the Mental Model of Computer Graphics

The object-oriented paradigm defines *objects* as the integration of data and functionality. As shown in [Wis90] these objects match the mental model of the graphics programmer in a high degree. This means:

- Graphical representations can be seen as entities or objects. They may be manipulated and attributed as wholes.
- They have their own attributes which influence the object and nothing else[1].
- Attributes may directly be assigned to or queried from the object itself (principle of locality). That does not mean, that objects cannot share physically existent attributes as shown in [Bei94c].
- Operations can be applied directly to the results of interactions (e.g. a pick operation). This is known as symmetry between input and output [Wis90].

Conventional systems often violate the principles of locality and input-output symmetry. For instance, in PHIGS (PLUS) [ISO89, ISO92] the attributes that are valid for a given structure cannot be queried because the temporary results of the traversing process cannot be accessed. The pick operation in PHIGS delivers a so-called pick path, from which the picked output object has to be extracted. Assigning an attribute to the picked object may cause fatal results in the central structure store because this structure may be multiple referenced and an automatic individualization is not provided in PHIGS.

[1] An exception is the inheritance of attributes inside a part-of hierarchy

1.2 Managing the Complexity

Using *inheritance*, a very high level of abstraction of the (application-defined) graphical primitives can be obtained. This includes not only graphical output functionality, but also semantics. Traditional systems that are based on display lists like IRIS GL [GL91] and PHIGS also allow the creation of arbitrarily complex output structures by recursive nesting of display lists (building a directed acyclic graph). However, these structures can not have an internal semantics and their parameterization is limited to the external assignment of attributes. Internal semantics and arbitrary parameterizability are useful:

- to implement dependencies from external conditions like modeling in space and time,

  For instance, a primitive *car* can rotate its wheels dependent on its forward/backward motion. A data glyph can change its visual representation in accordance with its new location and orientation in the data field [NBB94].
- to define abstract methods that hide internal structure,

  A primitive *house* may offer a method *openDoor* <*angle*> that hides the primitives and the actual modeling of the doors representation.
- to apply the assignment of (abstract) attributes [Bei94c] in a specific way and

  For instance, the assignment of an attribute *color* to a primitive *pushbutton* may apply the specified color to the pushbutton's foreground and the inverse color to the simulated shadow and to the text label.

- for a user-specific parameterization of structures (i.e. *classes*).

  As an example, for a class *WallWithDoor* a constructor parameter may specify the distance of the door from the walls origin.

1.3 Separation between Modeling and Rendering

This separation is useful from the view of software engineering. It allows for construction of customized modeling packages on top of a predefined, more or less generic and extensible graphics kernel. On the other hand, a single graphical scene can be rendered in various ways, e.g. producing a real time low-quality output based on a local shading model (e.g. wireframe or Gouraud shading [Gou71]) or a high-quality output basing on a global illumination model like ray-tracing [App68] or radiosity [GTGB84]. Ray-tracing is very time-intensive; an entire recomputation[2] must be done for each frame. Radiosity can be very fast for static scenes; if the distribution of the light is computed, frames can be generated as fast as in Gouraud shading.

In practice there is a need for both extremes. Object-oriented systems can provide different kinds of rendering in parallel and in a very consistent manner. This is useful to create animations using the fastest renderer and afterwards computing the same frame sequence in ray-tracing mode. Composite or user-defined primitives behave equally, independent of the type of rendering used. An elegant way to integrate modeling and rendering interfaces[3] for a given class is to use multiple inheritance.

In contrast, most traditional graphics systems (ray-tracing packages, ISO and industrial shading libraries) are strongly bound to one kind of rendering and are not extensible, in either modeling or rendering. Thus, wireframe modelers are used to create scenes that will be rendered later with ray-tracers like POV [pov92] in a batch process.

1.4 Interaction

Object-oriented mechanisms can also be the basis for an extensible and highly customizable interaction concept. This means:

- New interaction classes may be implemented which have a semantics that is application-dependent and can be much more complex in comparison to conventional interaction classes such as *pick* and *locator*.
- New and advanced physical input devices can be represented by extended interaction classes.
- Filters can realize a flexible mapping from different interaction classes to a single interaction class.
- Composite mechanisms may increase the level of interaction by providing more complex interaction classes that hide their implementation (low-level interaction objects).
- The feedback may be realized in a user-defined way. Even user-defined primitives can customize the feedback.
- In accordance with the above mentioned control of graphical output an immediate coupling between input and output in accordance with the paradigm of external control may be realized.

In contrast, traditional systems are often based on the principle of internal control (managing an exclusive and central event-loop) and define a canonical set of interaction classes and feedback variants, e.g. the GKS input model [ISO85].

1.5 State of the Art

Pioneer work in the area of object-oriented graphics has done with the GEO++ system [Wis90]. This includes mainly the subjects attributes, part-of hierarchies and handling of multiple references. Though this system is often cited, its concepts are unfortunately not subsequently considered in the design of other systems. GEO++ simulates the principal GKS/PHIGS functionality on top of the Smalltalk [GR89] graphics kernel.

Several class libraries encapsulate conventional shading libraries, e.g. GROOP [KW93] which is built on top of IRIS/AIX GL, and provide additional functionality (e.g. animation support).

In contrast, other systems are independent from a particular illumination model. They allow the rendering of a graphical scene with different renderers. Examples are the AVS graphics kernel Dóre [Kap91], GRAMS [EK92] and YART. An upcoming ISO standard on this area is PREMO [H+94, ISO94].

YART is a general purpose 3D graphics kernel, that supports various kinds of rendering and is ported to multiple platforms, e.g. OpenGL [NDW93], IRIS GL, PHIGS PLUS and X11 [Nye90]. YART also contains an extensible interaction model and offers an interpretative language binding (Tcl [Ous91b]) that is compatible with the C++ [ES90] API.

In the following we want to consider some design and implementation concepts embodied in YART which make this kernel useful for a heterogeneous, network-distributed virtual reality environment.

2 Graphical Output

2.1 Independence of Shading Model and Low-Level Library

As seen from the API programmers view, the YART graphical primitives are independent from the shad-

[2] with exception of scene subdivisions

[3] sets of methods for the communication between the primitives and modelers or renderers

ing library underlying YART and even from the implemented kinds of rendering. As mentioned above different protocols for communication with renderers (and modelers) can be integrated into the YART base primitives via multiple inheritance. YART currently supports {wireframe, Gouraud} shading, ray-tracing and non-ideal diffuse radiosity. Fig. 1 depicts the integration of rendering and modeling interfaces in the YART primitives. Additionally a perspective camera is provided that implements these illumination models. This camera can be switched from one model to another simply by calling one method.

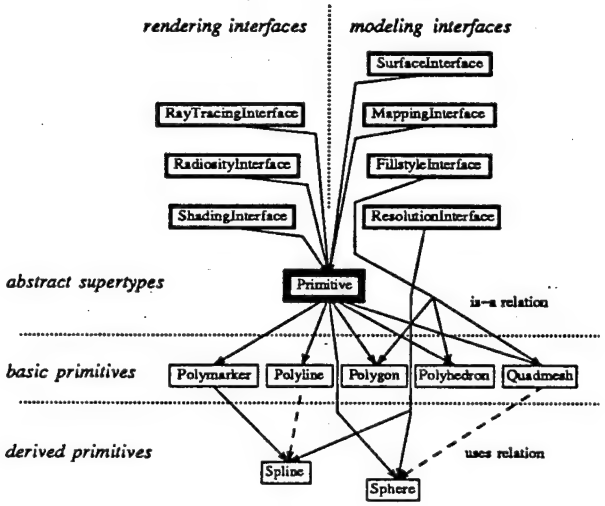


Figure 1: Topical configuration of the YART graphics kernel.

Thus, on different platforms specialized illumination models can be used for the same scene description. A highly parallel system may permanently run in ray-tracing mode, while most platforms (PCs, workstations) use a shading mode. In the future, new illumination models can be integrated into the general YART distribution or in special implementations.

To maximize portability YART uses a low-level API as the interface to platform-dependent low-level graphics, interaction and window handling routines. Using this API, ports to various platforms could be implemented with little effort. Less than 5% of the current YART code is platform-dependent and this portion will decrease in parallel with the implementation of high-level facilities such as advanced modeling classes and parametric curves and surfaces.

2.2 Transfer of High-Level Primitives

In contrast to most traditional shading libraries which define universal primitives such as polyline, polygon and quadmesh, the YART graphics kernel supports analytical and other high-level primitives[4]. Examples for analytically defined primitives are sphere, cylinder, cone, torus and so on; other high-level primitives will be mostly constructed by reading an external data file. Examples are Hershey [Her67] or TrueType [MS92] 3D text, OFF [Ros89] based primitives or primitives constructed from scientific data sets. In most cases it is much more efficient to transfer just these high-level primitives instead of their tesselated low-level representations. For instance, a sphere is created in YART by a string like *Sphere s 1.3* plus some additional modeling[5] (e.g. *s -ambient { 1 0 0 }*). To transfer this sphere via the DGL protocol (distributed GL [GL91]), for instance, a mesh in the current resolution of the sphere (e.g. 40x40 vertices) has to be sent over the net including vertex normals and colors.

Additionally, for often used data sets, such as fonts or OFF data, the transfer of the data sets can be avoided because they may exist both on the server and client sides.

2.3 Efficient Modeling

One design aim of the YART kernel was to minimize the modeling data to the absolutely minimal amount of memory. Inside hierarchies of graphical primitives the geometrical modeling and attributes will be inherited (hierarchical modeling). This makes the programming easy. In addition by using references and automatic individualization of attributes and modeling matrices a lot of memory can be saved. This also reduces network costs, because redundant matrices and attributes do not exist explicitly[6] and therefore, do not have to be sent via the net.

2.4 Abstract Attribute Types

YART provides abstract attributes in the sense that these attributes do not belong to a given class of a primitive. Every primitive can implement the assignment of a specific attribute in its own way. Currently, YART supports following abstract attribute types:

- The *resolution* attribute allows a primitive to tesselate itself dynamically. This is used for the mapping of analytical primitives and parametric curves and surfaces[7]. A high resolution implies a good approximation of the mathematical shape and a higher quality for the Gouraud shading or the radiosity computation.
- The *fillstyle* attribute switches a primitive into different representations. Currently *wireframe* and *solid* representation are supported as well as some bounding box styles.
- The *surface* attribute contains several surface parameters such as diffuse and specular coefficients. Others are transparency, refraction and emission.

[4] also supported by other object-oriented systems or (conventional) ray-tracers

[5] see next section

[6] They are expressed implicitly in the analytic primitives.

[7] A YART extension with these classes is in development but not released yet.

- The *mapping* attribute specifies extended surface parameter, e.g. a solid texture, an image or a bump mapping. The mapping of this attribute to the representation of the primitive is currently only available in ray-tracing mode and not for all primitives.

The rendering time for a given scene is very dependent on the first attribute types described above. Assuming that these attributes are **not** set for specific objects, the (global) default attributes will be used. Thus, on different platforms a specific output can be realized by just setting the default attributes, e.g. in an initialization file:

AttributeObject -fillstyle 1 -resolution 0.33

This turns on the solid fillstyle and sets the resolution to 0.33. A setup like this could be used on a Silicon Graphics machine, which has a good shading performance. On platforms where our X11 Gouraud shader[8] must be used in absence of a hardware-based shader one could use the following setup:

AttributeObject -fillstyle 0 -resolution 0.1

This turns on the wireframe display and a minimal representation of analytical primitives for all graphical objects that do not override these attributes.

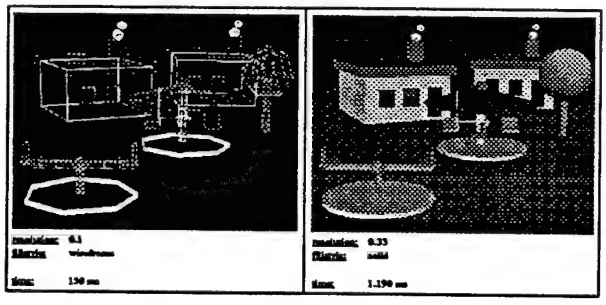


Figure 2: A scene rendered with different values for fillstyle and resolution.

Figure 2 shows a YART scene with different values of attribute types *Fillstyle* and *Resolution*. The rendering was done on an entry-level SGI workstation (R3000 IRIS Indigo) using the IRIS GL interface of YART. The time is the average of about 100 rendering processes. Each rendering process performs a complete update of attribute references and a propagation of attribute changes.

The resolution attribute may be synchronized with the motion velocity in a scene. For a fast motion in a scene a very low resolution should be used. However, if the motion stops for some time, the resolution could be increased. An advanced feature is to decrease the resolution of objects which are far away from the camera. This saves memory and rendering time.

As described in [Bei94c] new attribute types can be implemented in YART in the same manner as these predefined ones, e.g. a *text* attribute containing the font and a vertical scaling value. The new attributes will be handled with the same efficiency.

3 Interaction

YART implements a highly flexible and extensible interaction model [BS94], that is based on an extensible set of event types and event consuming input objects with an optional hierarchical structure.

The focus of this model lies on events, not on interaction classes. There exists a set of event types, which includes predefined event types for basic interactions like mouse and spaceball control. Higher level events may be derived from these or may correspond to advanced interactions. The orientation in portable basic events, which have to be implemented in dependence of the library below YART (e.g. IRIS GL or X11), allows the definition of interaction classes in a portable and flexible way.

Besides the events there exists a set of input objects at run-time. An input object is defined as follows:

- It consumes a (potentially empty) set of event types.
- It generates a (potentially empty) set of event types.
- It is defined by a state that includes the last occured event(s).
- There is a trigger defined; when triggering of the input object a list of callback objects will be invoked.
- It may have a father object. This may be a renderer object if the interpretation or generation of events depends on the renderer parameters (e.g. viewing transformation). The father object may also be another input object, allowing the definition of composite input objects with a higher abstraction level. Of course, input objects may be driven in stand-alone mode. Additionally, each input object may have an arbitrary number of children.
- It generates a visible feedback dependent of its state. Even user-defined primitives may be used to create the feedback.

Several interaction classes have been implemented, some for demo purposes, others to provide a minimal functionality. In every case, they are examples and can be replaced by user-defined ones.

- *Pick* and *(3D-)Locator* realize functionality compatible to interaction classes of the GKS/PHIGS input model.
- The *Manipulator* allows intuitive[9] manipulations of primitives in six degrees of freedom and supports both mouse and spaceball.
- The *FileDevice* provides support for file descriptors, such as sockets. Using a special

[8] which is actually very slow compared to IRIS GL on entry-level SGI

[9] i.e. in accordance to the user's expectation independently of view orientation and local and inherited modeling

parameterization[10] the stdin/stdout streams can be used to control applications via textual commands in parallel with other interaction objects, e.g. WYSIWYG user interfaces.

Others are vertex manipulator for vertex-based primitives or a walk-thru object for VR applications that maps mouse/spaceball motions to the camera parameters (view point, reference point). Future event types and interaction classes may encapsulate more advanced, real-word actions.

4 Interpretative Language Binding

Beside the C++ API, which is a result of the implementation, YART also offers an interpretative language binding that is consistent to the C++ one. This binding and the related object model is described in [Bei94e]; as its basis, the extensible {C, LISP, UNIX-Shell}-like interpreter language Tcl is used.

4.1 Interfacing with Tcl

Tcl is a type less interpretative language that operates on strings. Tcl commands are sets of strings. The first string identifies the command, the remainders are taken as arguments and will be evaluated by the command. Consequently, object-oriented techniques such as method calling can be realized easyly, using this flexible approach.

In opposite to many interpretative languages, the aim of Tcl is to be extended by application-specific commands. Creating a Tcl interface to an existing C/C++ application consumes some additional effort[11], but provides some important features:

- There are no turn-around times[12] when using the Tcl API. This is useful for learning and debugging purposes, and for prototyping.
- The definition of the Tcl interface, including the mapping of C(++) data structures to Tcl variables and values, normally forces the specification of a clean C(++) API. In most cases this API provides a high abstraction level hiding lower-level data and functional structures, which can be implemented efficiently in C or C++[13].
- Once defined, the Tcl API may be used as textual interface for user input, but also to put a WYSIWYG user interface on top or to have a network-wide access to the C++ objects. The user interface aspect is very important, because a strong separation between application functionality and extensible user interfaces is provided. Additionally, Tcl comes with Tk [Ous91a], which is a widget toolkit similar to OSF/Motif, but accessible thru Tcl.
- Furthermore, the definition/implementation of scene description or metafile formats is easy when using the Tcl API.
- Advanced features of such a Tcl/C++ mapping [Bei94b] may include the interactive behavior query of the C++ objects and the automatic creation of user interface dialog boxes or manual pages.

For a network-based virtual reality environment following facts can be extracted from the abovementioned:

- persistent storage of graphical scenes (generating calls to the Tcl API) and
- network-transparent and platform-independent access to C++ objects (transparent execution of Tcl API calls via the net).

The first fact makes it possible to get the state of a scene just by generating a scene dump. The second feature allows the transfer of complete scene dumps or incremental scene changes via the net - without consideration of binary formats and byte sequence.

4.2 Interpretative Class System

On top of the Tcl API an interpretative and persistent class system is implemented [Bei94e, Bei94b]. This class system allows the definition of new classes (e.g. primitives) at run time in a portable way. Thus, it is possible to increase the abstraction level by defining more complex primitives. However, a more important point is, that semantics can be integrated in these classes (by defining methods) in a portable (and even network-transparent) way. Conventional scene description languages do not offer this possibility.

The class system supports single inheritance, public methods and private members. It is compatible to the general YART C++/Tcl object model and offers persistency, too.

4.3 On the Way to a VRML

Basing on this interpretative language binding an interactive VR Makeup Language could be specified by:

- a definition of a subset of Tcl (no use of *exec*, *proc*, *pwd*, *cd*, etc.)[14]
- a definition of the YART API including the class system
- a definition of specific YART extensions - so-called *packages*, e.g. for scientific visualization
- a definition of mechanisms to provide data sets (separate transfers, load paths, etc.)

[10] fd 0 for stdin

[11] However, this can be minimized using object-oriented techniques as shown in [Bei94e]. The author evenly implemented an 1:1 Tcl interface to OpenGL in a very generic and efficient way by parsing the OpenGL header files and using implicit C++ operators for parameter conversion.

[12] edit-compile-link-run/debug cycles

[13] Numerous projects stand for the usability of this design philosophy, such as a commercial CFD system, a virtual reality application and a 3D graphics kernel.

[14] These Tcl procedures would provoke incompatible scene descriptions.

5 A Concrete Implementation - YART/VR

The package presented in the following is more an application of the YART graphics kernel than a serious VR tool. Nevertheless it realizes some concepts which could be interesting in this context.

5.1 Metaphores

The application is based on two metaphores: the *scene* and the *user*. A scene is a kind of a server running anywhere in the net and can be addressed via <scene_name>@<host_name>, e.g. castle@141.24.32.29[15]

A physical user is represented by an object of a Tcl class that can be specified freely by the user. The representation is built from YART primitives like sphere, block, cylinder, quadmesh, polygon, polyhedron, text and so on. Additionally, specific methods can be provided for the user class.

The scene currently accepts following user commands:

- login/logout
- move forward (backward) by a specified distance in view direction
- move upward (downward) by a specified distance (i.e. in y direction)
- rotate left/right by a specified angle
- send a pick ray into the scene
- broadcast a textual message to all users
- call a method of the user representation with an arbitrary number of arguments

User commands[16] will be sent to the scene and the scene sends resulting commands back. Thus, the scene controls all user movements and can prevent a user from going thru a wall, for instance.

Scenes and users correspond to a client-server architecture. The user application is the client which contains a hidden copy of the global and visible scene. All changes result in the broadcast of small-sized sequences of Tcl commands (typically less than 500 bytes) to all mirror scenes. Thus, YART/VR implements incremental scene changes. Of course, for the initial setup of a mirror scene the topical state of the server scene must be transferred over the net. This scene dump is typically 100 kBytes and more; however, the transfer costs can be reduced by using a compression utility.

5.2 Implementation

Two executables are provided, when YART/VR is installed. These are standard tools and can be replaced by other ones.

vr_scene will be called by the executable scene description files. Principally, scene description files are standard YART metafiles, which can be created/modified using a text editor or a direct-manipulative user interface. Fig. 3 shows a screen shot of a scene representing our institute; the related scene description file has a size of 20 kbytes well-formatted and documented Tcl code. By the way, this scene consists of about 6.700 YART primitives.

Additionally, in a scene file may be specified: the initial position and view orientation of a user that comes into the scene and the maximal number of users allowed in the scene.

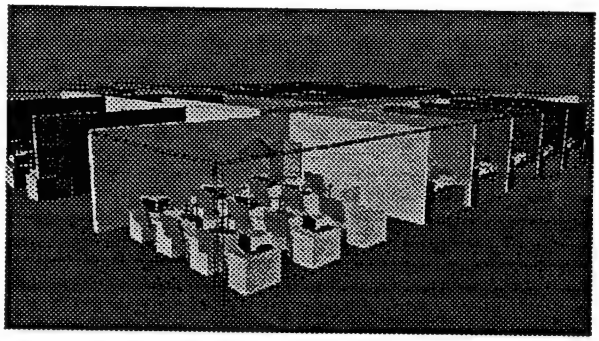


Figure 3: A scene representing our institute.

The predefined semantics of a pick in the scene is as follows: If a user was hit by the ray the picking user gets a message containing the picked users name. The picked user will be informed that he was picked by the picking user. This semantics may be overwritten in a scene file. Its implementation may use commands for transferring of binary large objects and for sending of messages or commands.

vr_user is a Tk based front end (fig. 4) that provides mouse-driven interactions for moving in the scene, scales for camera parameters and text input/output. Per convention, *vr_user* expects a file *${HOME}/.yartvr* that contains the user's representation in form of a class definition (and implementation) and supports a file *${HOME}/.vr_userrc* for configuration of the front end (see fig. 2).

The current implementation of scene-user communication is based on internet domain sockets. As platforms all UNIX systems with at least X11 graphics and BSD compatible networking are supported.

5.3 Extensions

Both client and server may be extended without loosing compatibility to running applications.

5.3.1 User (Front Ends)

The front end (currently the *vr_user* program) may be extended in following ways:

- stereo output for SGI machines (a YART interface to SGI stereo hardware and IRIS GL is part

[15] This host isn't reachable via the net.

[16] These are on a high level, hiding elementary mouse and spaceball motions, and result partially from precomputations on client-side, such as pick-correlation.

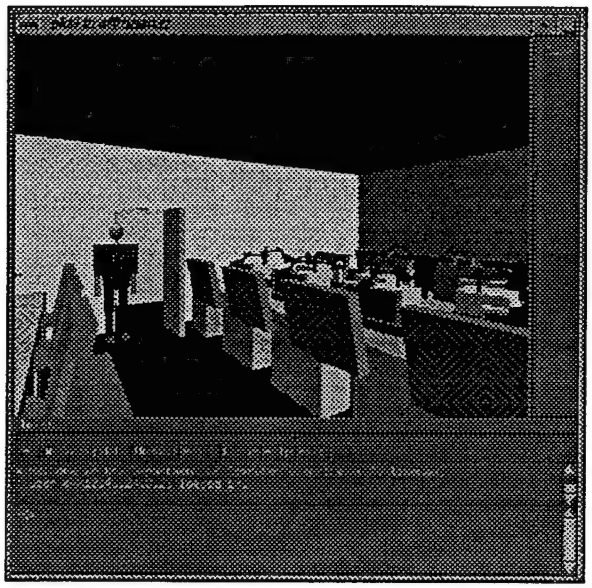


Figure 4: Meeting myself in a room of our institute.

of the YART distribution),

- textual control of movement allowing faster and more precise motion,
- radiosity and ray-tracing images of the scene to capture an impression in an advanced rendering quality and
- spaceball integration (YART already provides spaceball events for that.).

5.3.2 Scene

The complete semantics of user motion and picking is controlled by the scene. Thus, advanced scenes (subclasses of the predefined scene class) may implement any specific behavior. An intelligent handling of subscenes may reduce the rendering and transfer time drastically.

5.4 Communication

The current socket/XDR [KP84] stream based communication is nothing other than a subclassed realization of the abstract communication classes. Thus, other implementations (e.g. usage of shared memory in local connections) could be used for future versions. Additionally, the usage of the GNU compressor *gzip* for the transfer of BLOBS is hidden inside the concrete communication classes.

6 Future Research Subjects

The following topics are taken from a proposal for a project sponsored by the German research community. The aim is to extend the YART graphics kernel for the special needs of complex VR scenes.

6.1 Scalable Rendering Technology

Up to now, there is a strong separation between local and global illumination models and related kinds of rendering. Most packages implement exactly on model and sometimes a second one for pre-viewing. YART integrates both local and global illumination models and is open for new ones. However, the different kinds of rendering are strongly separated in YART. Is a scalable rendering technology realizable that integrates existing and new kinds of rendering and allows a linear increasing rendering quality?

6.2 Time-Dependent Rendering

Normally the rendering time t is a function of rendering quality q and the scene complexity c. For a given (q, c) on a specific platform a certain amount of time will needed to render the scene. An interesting question is: Is it possible to choose q and c in dependence of a predefined time value?

6.3 High-Level Clipping and Data Management

The rendering time of ultra complex scenes can be reduced by using some kinds of high-level clipping to exclude invisible (hierarchies of) objects much earlier from the rendering. A solution for this under the condition of rectangular elements (e.g. rooms in a house) is proposed in [Bro86].

The management of the parallel existent objects is somewhat equivalent to the former point. Objects which are invisible and do not have an own semantics don't need to exist and therefore can be removed from the memory.

7 Conclusions

YART/VR is based on

- Tcl - an extensible interpreter language
- Tk - an X11 widget set based on Tcl
- YART - a general-purpose 3D graphics kernel
- IOM - Tk based tools for programming YART

All these packages are public domain and can be ftped from *ftp://metallica.prakinf.tu-ilmenau.de/-pub/PROJECTS/GOOD*.***[17] (or from other ftp sites). The directory contains a *README* file that describes all the files in this directory.

A set of WWW documents about YART, IOM, YART/VR, etc. can be reached via *http://metallica.-prakinf.tu-ilmenau.de/GOOD.html.*

The author is available on *institute@speedy.prakinf.tu-ilmenau.de* when he is at the office and would be happy to receive visitors from the net.

Special thanks goes to Frank Wicht, Jochen Pohl (both TU Ilmenau) and P. Bryan Heidorn (University of Pittsburgh) for supporting this work.

[17] Look for the highest number, please.

References

[App68] A. Appel. Some techniques for shading machine renderings of solids. 32:37–45, 1968. AFIPS 1968 Spring Joint Computer Conference.

[Bei94a] E. Beier. Design and Aims of the YART Graphics Kernel. *RTNews*, 7(3), 1994.

[Bei94b] E. Beier. Interpretatives Klassensystem fuer Tcl mit C++. *iX Multiuser Multitasking Magazin*, pages 148–152, January 1994.

[Bei94c] E. Beier. Object-Oriented Design of Graphical Attributes. In *[Eur94]*, pages 41–50, 1994.

[Bei94d] E. Beier. *Objektorientierte 3D-Grafik.* Int'l Thomson Publishing, September 1994.

[Bei94e] E. Beier. Tcl meets 3D - Interpretative Access to Object-Oriented Graphics. In *2nd Tcl/Tk Workshop, New Orleans*, pages 159–170, June 1994.

[Bro86] F. P. Brooks. Walkthrough - A Dynamic Graphics System for Simulating Virtual Buildings. In *Workshop in Interactive 3D Graphics.* ACM, New York, 1986.

[BS94] E. Beier and R. Schachtschabel. An event based and hierarchical interaction model for PREMO. Technical report, 1994. ISO/IEC PREMO Working Draft, Part 3, SC24/WG6 Meeting, Amsterdam, 1994.

[BW91] E. H. Blake and P. Wisskirchen, editors. *Advances in Object-Oriented Graphics I.* EuroGraphics Seminars, Springer-Verlag, 1991.

[EK92] P. K. Egbert and W. J. Kubitz. Application Graphics Modeling Support Through Object Orientation. *IEEE Transactions on Computers*, pages 81–92, October 1992.

[ES90] M. A. Ellis and B. Stroustrup. *The Annotated C++ Reference Manual.* Addison Wesley, 1990.

[Eur94] EuroGraphics Seminars. *4th EuroGraphics Workshop on Object-Oriented Graphics*, 1994. Sintra, Portugal.

[GL91] Graphics Library Programming Guide. Technical Report 007-1210-040, Silicon Graphics Inc., 1991.

[Gou71] H. Gouraud. Computer Display of Curved Surfaces. *IEEE Transactions on Computers*, 20(6):623–628, 1971.

[GR89] A. Goldberg and D. Robson. *Smalltalk-80. The Language* . Addison Wesley, 1989.

[GTGB84] C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile. Modeling the Interaction of Light Between Diffuse Surfaces. *Computer & Graphics*, 18(3):213–222, 1984.

[H+94] I. Herman et al. PREMO - An ISO Standard for a Presentation Environment for Multimedia Objects. In *Proceedings of the '94 ACM Multimedia Conference, San Francisco, CA.*, 1994.

[Her67] A. V. Hershey. Calligraphy for Computers. Technical Report No. 2101, U.S. Naval Weapons Laboratory, Dahlgren, 1967.

[ISO85] ISO. Information Processing Systems – Computer Graphics – Graphical Kernel System (GKS). Technical Report ISO 7942:1985, International Organization of Standardization, 1985.

[ISO89] ISO. Information Processing Systems – Computer Graphics – Programmer's Hierarchical Interactive Graphics System (PHIGS). Technical Report ISO/IEC 9592:1989, International Organization of Standardization, 1989. Parts 1 – 3.

[ISO92] ISO. Information Processing Systems – Computer Graphics – Programmer's Hierarchical Interactive Graphics System (PHIGS), Plus Lumière Und Surfaces. Technical Report ISO/IEC 9592-4:1992, International Organization of Standardization, 1992. Part 4.

[ISO94] ISO. PREMO - Presentation Environment for Multimedia Objects. Technical report, International Organization of Standardization, 1994. Initial Draft 1.5.

[Kap91] M. R. Kaplan. The Design of the Dóre Graphics System. In *[BW91]*, pages 177–198. 1991.

[KP84] B. W. Kernighan and R. Pike. *The UNIX Programming Environment.* Prentice-Hall, 1984.

[KW93] L. Koved and W. L. Wooten. GROOP: An Object-Oriented Toolkit for Animated 3D Graphics. In *OOPSLA'93*, pages 309–325. ACM, 1993.

[MS92] Microsoft TrueType Font User Manual. Technical report, Microsoft Corporation, 1992.

[NBB94] J. Nuetzel, E. Beier, and R. Boese. Scientific Data Exploration Using Multiple Configurable Data Glyphs. In *5th EuroGraphics Workshop on Visualization in Scientific Computing.* EuroGraphics Seminars, 1994. Rostock.

[NDW93] J. Neider, T. Davis, and M. Woo. *OpenGL Programming Guide.* Addison Wesley, 1993.

[Nye90] A. Nye. *Xlib Programming Manual.* The Definite Guides to the X Window System. O'Reilly & Associates Inc., 1990.

[Ous91a] J. K. Ousterhout. An X11 Toolkit based on the Tcl Language. In *Proceedings to 1991 Winter USENIX Conference*, 1991.

[Ous91b] J. K. Ousterhout. Tcl: An Embeddable Command Language. In *Proceedings to 1991 Winter USENIX Conference*, 1991.

[pov92] Persistence of Vision Ray Tracer (POV-Ray), Version 1.0. Technical report, July 1992.

[Ros89] R. J. Rost. OFF - A 3D Object File Format. Technical report, 1989.

[Wis90] P. Wisskirchen. *Object-Oriented Graphics - From GKS and PHIGS to Object-Oriented Systems.* Springer-Verlag, 1990.

A Fuzzy Controlled Rendering System for Virtual Reality Systems Optimised by Genetic Algorithms

PROF. DR. R. D. SCHRAFT, J. NEUGEBAUER, T. FLAIG AND R. DÄINGHAUS.

Fraunhofer-Institute for Manufacturing Engineering and Automation (IPA), Nobelstaße 12,
D-70569 Stuttgart, Tel +49 711 970 1308,
Fax +49 711 970 1005, e-mail: daeinghaus@ipa.fhg.de.

Abstract - *This paper describes a new rendering tool for the fast and efficient rendering of virtual environments. The usage of rendering modules with dynamic level-of-detail technology is the upcoming answer to the up to now insufficient graphical hardware. The paper outlines the design of a fuzzy-based dynamic level-of-detail controller and optimisation with genetic algorithms. Furthermore the potential of this new technology will be shown with the summary of recent development work at the Fraunhofer-Institute for Manufacturing Engineering and Automation, which has made possible the use of Virtual Reality for robot simulation, off-line programming and remote operation of industrial robots*

Keywords: Virtual Reality, Fuzzy-Logic, Genetic Algorithms, Robot Simulation.

1. INTRODUCTION

International competetiveness is characterised by the reduction of innovation time. Therefore the success of new products strongly depends on the necessary time for their development. Virtual Reality, as a new 3D human-computer interface significantly accelerates the processes of creating and handling 3D data in 3D space for design and evaluation purposes.

Virtual Reality, providing advanced human-computer-interfaces can make a valuable contribution to improve simulation and control tools. The operator, using a dataglove and head-mounted stereo display can act in a virtual world and is no longer a passive observer [1].

Characteristic in Virtual Realtiy is the interaction and the perception in a Virtual environment. This is only possible by real-time simulation and rendering. This led to expensive but high performance computer technology.

The Fraunhofer-Institutes carrying out applied research for small and medium-sized enterprises (SME) is familiar with industrial demands. For establishing VR simulation technologies in industry it is necessary to develop fast rendering algorithms to be able to use relatively cheap hardware for real time rendering tasks.

This paper outlines a new dynamic level-of-detail algorithm which is based on a fuzzy-controller with parameters which are optimized by genetic algorithms.

2. REQUIREMENTS FOR VISUALISATION TOOLS

All graphical systems have limited capa-

bilities that affect the number of geometric primitives that can be displayed per frame at a specific frame rate. The general requirement is a maximum frame rate including optimal quality of the displayed scene. These two goals are in contradiction to each other. The driving force behind IPA development is to find the best compromise between quality of the scene and frame rate.

Because of the existing limitations, maximising visual output quality while minimising the polygon count, level-of-detail processing is one of the most promissing tools available for managing scene complexity for the purpose of improving display performance.

To reduce rendering time, objects that are visually less important in a frame can be rendered with less detail or in a lower degree of abstraction. The level-of-detail approach to optimising the display of complex objects is to construct a number of progressively simpler versions of an object and to select one of them for display for example as a function of range.

This method requires the creation of multiple representations of an object with varying levels of detail or levels of abstraction. Rules must be given to determine the best representation of the object to be displayed [2].

The requirement must be met, that enough representations of an object with different quality levels exists. Only on a comprehensive list of graphical representations a satisfactory work of a graphical output controlled rendering is guaranteed.

To use extended level-of-detail-technology, a large set of graphical representations must be designed. Several methods are known. The easiest method is to design repesentations in different degrees of abstraction by hand. Also semi automatic or automatic methods are possible. The objective is to get representations with different numbers of polygons. The rendering of any single polygon is the most important criterion for the graphical performance of the computer.

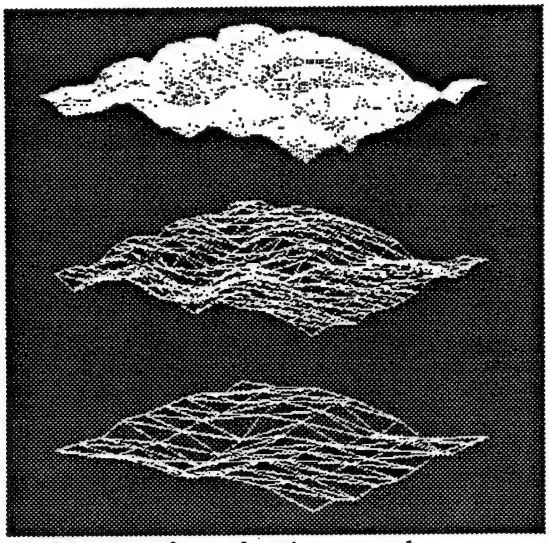
One very simple representation of an object is the bounding box, build up with six polygons, another one is a minimal convex hull. This representation looks like a rubber balloon stretched over the described object. Often it is enough to use this representation in a very complex scene or in case of background scenery.

Other automatically created level-of-detail representations can be generated by defining tolerance measures around the surface of the described object. If neighboring vertices are inside of this predefined tolerance measure, all polygons belonging to these vertices are merged. With this method different level-of-detail representations can be designed by graduated tolerances. This procedure is very efficient to design large landscapes, Figure 1.

If the virtual world is designed by polygonal representations of analytical surfaces like cylindrical, torus or cone surfaces, each surface can be directly created in different representations. These must be fine-tuned, because non-fitting adjacent surfacees could result. A broomstick for example could be shown by a cylinder with a rounded end built of a spherical surface. The usage of different levels of detail will create a confusing look. There is the possibility of holes appearing and edges might not fit anymore, Figure 2.

Assuming that enough different representations of any object are designed, the algorithm deciding the current representation is the most important part of the rendering modul. Rules have to be defined to change the level of detail. The

governing influencing factors are the following [3]:



1.) Distance from the viewer to the

Figure 1: Three landscape representations

concerned object

The distance is an important parameter. The nearer an object is, the more important it is for the whole scene.

2.) Size of the object

Of the same importance is the size of an object. Both parameters together define the size of the considered object on the screen.

3.) Position in relation to the center of the display

The user wearing a Head Mounted Display will usually look in the center of the viewed scene. Objects on the edge of the visible frustrum are therefore assumed to be less important then objects in the middle of the screen.

4.) Complexity of the object

The complexity of an object is expressed by the number of polygons used to display the object.

5.) Special interest of the object

Some objects have a predefined special interest. In a robotic workcell the gripper of the robot e.g. would be more interesting than many other objects.

6.) Velocity of the object

Using velocity of objects as a parameter the importance of fast objects can be reduced as these can't be seen correctly, like for example the propeller of a plane.

A continuous change of the representations of different objects may result from free navigation of the user in the virtual world. This change can be a visual problem, because of the resulting tremendous flickering of the whole scene. As the human eye especially perceives motion a continuous change of object representations disturbs the impression of the viewed scene.

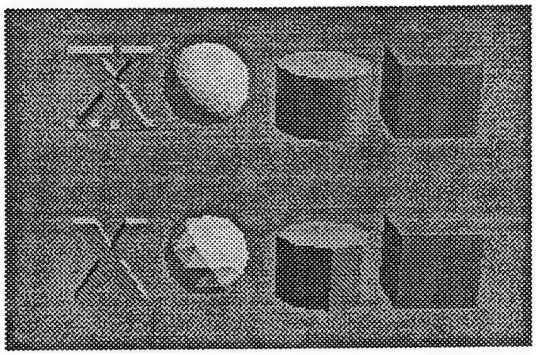


Figure 2: Arrangement of geo primitives

The problem in adjusting the rendering modul are the many parameters in the abstraction algorithms. The question is how to reach the optimum between high frame rate and high level of detail. One condition is that the visual output is done for a subjectivly deciding human being. Another is the contradiction because of opposite goals of the criteria. For example an object at a far distance is not very important, although the same object has a higher importance possibly because of a definition as an object of interest. The arising contradiction can be balanced by using fuzzy logic algorithms.

3. A DYNAMIC ADJUSTABLE RENDERING SYSTEM

The dynamic control of the graphical level-of-detail has various advantages and has become state of the art with the introduction of the Silicon Graphics rendering package Performer. The philosophy is to use graphical abstractions for some scene objects in a stress situation of the graphical output. But the only benefit of the SGI Performer is to decide either to give the best (normal) quality or to simplify the scene by leaving out some details of the scene. Different representations of objects are possible, but very complicate to model.

Performer is layed out for peaks of stress situations. The disappearance of objects in these situations cannot be accepted in most applications. A controller has to be layed out to control the scene permanently, because stress situations must be expected anytime. A permanent abstraction of the scene has to be possible.

This can be reached best by using a fuzzy logic controlled rendering modul. With the use of a controller, a frame rate can be fixed. The controllers task is to reach the desired frequency of frames per second by selecting one of the different representations of the single objects. It is the user's job to give the system the best compromise between the fastest display frame rate and the best quality of the output to tolerate. This adjustment strongly depends on the subjective quality perception of the viewer.

The successful aproach controlling principle in discrete environments is the philosophy of fuzzy logic. All incoming values can simply be rated. All factors of the output decision can be considered carefully. The fuzzy logic controller gives a soft and easy to control switching of the representations of graphical objects. The fuzzy logic controler gives the best way in rating the complexity of the current scene and offers a soft deviation of the desired frequency. The problem of the disturbing flickering from the oscillation in conventional controllers can be minimised.

The benefit of a fuzzy logic controller is the very easy usage of input values by using linguistic variables. These variables - also called membership functions - describe the state of every influencing parameter in form of a natural language. The distance to an object of the scene for example can be descibed in words like *very near, near, middle, far* or *very far*. Every condition is represented by a fuzzy set. They define the memberships of the conditions to an interval of discrete values. If e.g. the current distance is detected by the system, the fuzzy algorithm could recognize the fuzzy set *far* to 80 percent and *very far* to 40 percent. All other parameters like size, complexity etc. are evaluated similar.

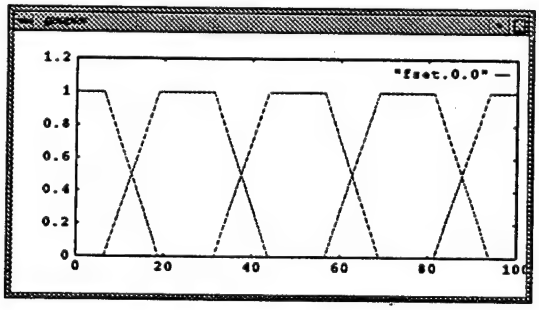


Figure 3: The membership function of the deciding parameters

All decisions of any parameter are brought together by rules. In these rules the connections between the different parameters (distance, size and complexity of objects etc.) of the scene description are evaluated like:

if the distance is far and the size is big then give a better representation

if the complexity is low and the size is small then give a more abstract representation

Any combinations of these rules are possible. The valid output of the rules can have opposite evaluations, but the important advantage of fuzzy logic controllers is the possibility to compensate all opposites and to find an optimal average in evaluating the input values.

The evaluation of the complete set of rules via defuzzyfication returns a discrete value, that the rendering modul uses to decide which representation is the current best choice. The more rules are used and the more variables are interpreted, the better is the stability of the controller. This stability is achieved by decreasing the controllers oscillation and the minimising of the withcoming flickering of the graphical output.

So far it is easy to define a controller to decide the representations of single objects. But the task is not to optimise the output of the objects in a scene, but the impression of the scene itself. The controller must be optimised to give the best subjective scene impression to the user.

The missing link to an improved controlled rendering modul is the evaluation of the complete scene. How can a complex scene be evaluated? The limited output ressources determine the decisions of the controller of every single object in the current scene. The evaluation of objects have to be seen in relation to the output of the whole scene. This means that the controllers decisions are depending not only on static parameters but on the connection of the scene building objects.

The fuzzy logic controller must be optimised by a training in well known virtual worlds. The decision base of the controller are the membership functions and the rule base. The rule base cannot be changed softly, all connections are fix. Size and the form of the fuzzy sets in the membership functions can be manipulated though.. The sets are defined as trapeziod shaped memberships, built up by a quadrupel The four coordinates defining the sets can be manipulated to get a modified characteristic of the controller.

How can the controller be adjusted to detect the best decision of object representatives with a minimum of oscillating but holding a fixed frame rate and an optimal subjective scene display? With the modification of all defining points of fuzzy sets in the five incoming membership functions 36 parameters can be found to modify. The best output depends on an optimum rating of these parameters.

4. OPTIMISATION BY GENETIC ALGORITHMS

Genetic algorithms (GA) have some properties that make them interesting as a technique for selecting high-performance membership functions for fuzzy logic controllers. Due to these properties, GAs differ fundamentally from more conventional search techniques. They consider a population of points and not a single one, they use non-deterministic probabilistic rules to guide their search. GAs consider many points from the search space simultaneously and therefore they have a reduced chance of converting to a local optima [4].

GAs work on the basis of the evolutionary theory [5]. The parameter set defining the problem to optimise - in the special case the defining points of the membership functions - are seen as a DNA of an biological individual.

The result of the search procedure is the definition of the optimal parameter set. This set can be found by starting the evolution with a large number of individuals with an identical parameter set (DNA). The DNA of individuals can be inherited. During the

inheritage the DNA of the individuals is modified by crossing or mutation. Crossing means the changing of random parts of the parameter set between two individuals, mutation is a mechanism that exchanges single parameters within the set of an individual. The descendants generated by these algorithms build up the new population.

The generated new individuals are tested iteration by iteration. The biological rule of the survival of the fittest is applied. Every new individual (parameter set for the solution of the optimisation problem) is rated in a typical environment.

The duration of these tests is to long to calculate thousands of iterations. Instead of this a average function of typical rendering outputs was created. This function simulates the typical characteristic of the rendering system, the rating of the single individuals can be calculated without graphical output. If the test has a positive result the current parameter set is a good solution for the membership functions. The individual is strong enough and keeps alife, otherwise the individual dies.

The best parameter sets for the membership functions can be found by following only the fittest individuals. Like in the evolution only the fittest individuals inherit their DNA. After a while the fittest individuals are selected and every parameter set will deliver a good definition of the considered membership function.

Genetic algorithms are not random walks through the search space. They use random choice efficiently in their exploitation of prior knowledge to locate

near-optimal solutions rapidly.

An example of the successful work of the manipulation of the membership functions can be seen in the Figure 3 and Figure 4.

The membership function for the distance to the current object is defined as

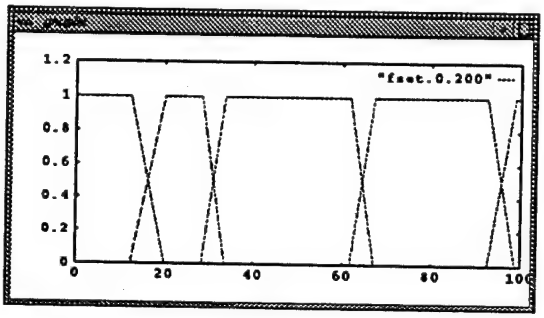


Figure 4: Optimised membership function

fuzzy sets with the conditions *very near, near, middle, far* and *very far*. The predefined membership function are defined symetrically to give the best assumption for the optimising genetic algorithm. The result of the optimisation can be seen in Figure 4.

The defined fuzzy sets still exists, the difference to the predefined membership function is the modified placement and shape of the single sets related to the complete function.

The rating algorithm is a simulated walk through a virtual world. A nonstatic change of visible objects is simulated by a step function response of the controller. In Figure 5 can be seen that the automatically optimised controller delivers a minimum oscillation and a very fast adjusting to the desired frame rate.

The rating algorithms are so fast that a learning on the fly can be reached. A permanent optimising in running simulations is possible. In every new virtual world the membership functions

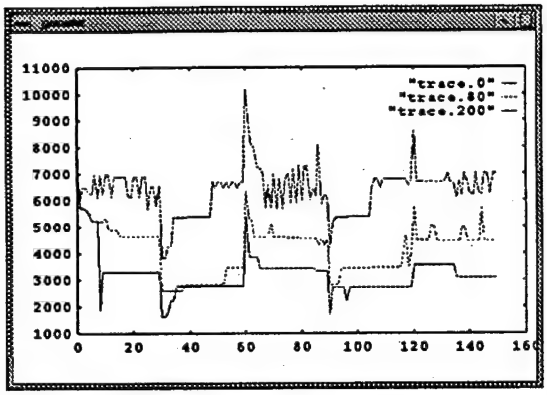


Figure 5: Frame rate of the controlled rendering modul

can be optimised. An individual controller parameter definition of the controller for every new scene can be designed [6][7].

5. VIRTUAL REALITY AT THE IPA

The IPA is at the forefront of Virtual Reality research in Germany since 1991. First industrial projects have been done in the field of manufacturing engineering [8]. Figure 6 shows an overview of the use of Virtual Reality system at the IPA. There are industrial projects in the planning of robot work cells, visualisation of CAD-data and presentation of product ideas. Further applications have been realised like off-line programming and teleoperation. Rapid Prototyping, manufacturing planning, assembly planning, operator systems and training systems are areas for further development.

6. THE DEMONSTRATION CENTRE VIRTUAL REALITY AT THE IPA

The high interest from the public and the industrial side was accorded to further Fraunhofer-Institutes which deal with this innovative technology. The Fraunhofer-Society has faced the challenge of this new technology and is now active in making this potential accessible to industry. The project is to be efficiently supported by the

Application Areas / Applications	Industrial Projects at the IPA	VR System at the IPA	VR System others
Robot Application Planning	●	●	
Robot Off-line Programming		●	
Robot Teleoperation		●	
Visualisation of CAD-Data	●	●	●
Presentation of Product Ideas		●	●
Presentation of Product Proposals	●	●	
Rapid Prototyping		○	○
Manufacturing Planning		○	
Handling- and Assembly Planning		○	
Operator Systems for Machines		○	
Interactive Learning Systems		○	○
○ concepts ● applications			

Figure 6: Virtual Reality Applications at the IPA

Demonstration Centre for Virtual Reality. In January 1993 this institution has taken up its work for the period of five years [9].

The establishment of the Fraunhofer-Institute Demonstration Centre for Virtual Reality is to present smaller and middle-sized companies new techniques of Virtual Reality, to reduce fear of contact between industrial practice and Virtual Reality research and to demonstrate prototypes of Virtual Reality applications in a practical and vivid way.

The following services are available in teh Demonstration Centre for Virtual Reality at the IPA in Stuttgart:

- Dispersion of availabel knowledge on Virtual Reality,

- Training of personnel from interested companies,
- Presentation of demonstration appliances and processes,
- Consultancy for companies,
- Application and test of new and already available applications and
- Development and demonstration of exemplary applications.

7. CONCLUSION

Virtual Reality technology makes a valuable contribution to improve simulation and control tools. The operator, using a dataglove and a head-mounted stereo display can act in a virtual world and is no longer a passive observer.

To get the best response from the Virtual Reality system, the graphical output must be efficiently high. Best usage of advanced rendering functionality (possible to combine with commercial products like Performer (SGI) or other high performance rendering systems) are reached by new developed control moduls.

As high performance can now be achived even with inexpensive hardware advanced rendering tools for VR systems can now be used for industrial applications.

8. REFERENCES

/1/ Thomas Flaig: *"Echtzeitorientierte interaktive Simulation mit VR4RobotS am Beispiel eines Industrieprojektes"*, Tagungsunterlagen des 2. VR-Forum '94, Anwendungen und Trends, Februar 1994, Stuttgart, Deutschland.

/2/ N.N.: *"IRIS Performer Programming Guide"*,1994

/3/ Ralf Däinghaus: *"Dynamische Geometriedatenhaltung für schnelles Rendern in effizienten Virtual Reality-Systemen"*, Tagungsunterlagen des 2. VR-Forum '94, Anwendungen und Trends, Februar 1994, Stuttgart, Deutschland.

/4/ Goldberg, D.E.:*"Genetic Algorithms in Search, Optimization and Machine Learning"*, Addison Wesley, 1989

/5/ Holland, J.:*"Adaption in Natural and Artifical Systems"*, The University of Michigan Press, Ann Arbor, 1975

/6/ Zadeh, L.A.:*"The Concept of a Linguistic Variable and its Application to Approximate Reasoning"*, New York, 1973

/7/ Karr, C. *"Genetic Algorithms for Fuzzy Controllers"*, AI Expert, P26-33, February 1991

/8/ J.-G. Neugebauer, T. Flaig: "IPA - Trendsetter der Produktionstechnik, Richtungsweisende Beispiele für wirtschaftliche Produktion und Automatisierung", MI Moderne Industrie, September 1993, Stuttgart.

/9/ J.-G. Neugebauer: *"Virtual Reality - The Demonstration Centre"*, Meckler Conference - Virtual Reality International '93, April 1993, London, Great Britain.

VIPER (VIrtuality Programming EnviRonment): A virtual reality applications design platform

Patrice TORGUET, René CAUBET

I.R.I.T. Université Paul Sabatier,
118, Route de Narbonne, 31062 TOULOUSE CEDEX, FRANCE
e-mail: torguet@irit.fr

Abstract: In this paper we present a new virtual reality applications design platform, VIPER, which emphasizes distributed aspects of virtual environments. Our goal is to propose the developer a generic system which can be specialized to suit an application, while minimizing his work all along the virtual environment definition. We describe the general structure of an environment in VIPER, and discuss the distribution of such an environment.

Keywords: virtual environments, application design platform, distribution, programming environment.

1. Introduction

In order to increase the realism of virtual environments, one solution is to allow many users as well as simulated entities to interact in a shared virtual environment. Such a virtual environment appears more real from each user viewpoint thanks to the rest of the environment liveliness. Applications and worlds in these environments are difficult to design and simulate mainly because of their distributed properties.

Our goal is to define a generic system for virtual reality application designing and virtual environment simulation. This system, called VIPER, has to simulate multiparticipant virtual worlds in real-time, thanks to a distributed platform.

In this paper we present the general structure of a virtual environment in VIPER as well as the framework of the system software architecture.

2. Survey of existing similar systems

Virtual environments operating systems can be roughly divided into two classes.

The first class is composed of systems dedicated to a certain type of application. Good examples of this class are: NPSNET [Zyda93] which is dedicated to military simulations and IPA's Transputer-based virtual reality workstation [Strommer93] which is dedicated to industrial robot control and programming. Those systems have developed domain specific accelerating techniques: specific hardware (dedicated Transputer architecture of IPA's VR workstation) and optimization algorithms (dead reckoning for NPSNET like systems). Those systems are very efficient for their specific domain but seem difficult to reuse for other applications.

Application generic systems compose the second class. Good examples of this class are: DIVE [Carlsson93], VEOS [Bricken93] and AVIARY [West92]. Those systems have to be the most generic possible in order to be easily used to design any application. This genericity, however, trades off with the general efficiency of the final application. For example in most of these systems, optimization algorithms are difficult to implement.

In order to better deal with efficiency within a generic system, we think that tools should be proposed to the virtual environment developer. Systems like WAVES [Kazman93a], have tried to implement optimizations in the core of the system. We rather think that the definition of such optimizations should be available for the virtual environment developer, either declaratively (the developer chooses an

available optimization for his current environment) or explicitly (the developer implements a new acceleration, eventually based on an existent one).

3. *Virtual environment structure*

VIPER is aimed for the design of every application that is based on a virtual environment which can be modelled using the following general structure.

In VIPER, a virtual environment is made of entities, stimuli and a virtual universe (Fig. 3.1).

- The *"entity"* paradigm allows uniform management of virtual worlds scenery, virtual objects and *"clones"*. Entities are autonomous and own a set of attributes and behaviours. They are conceptually grouped in *"families"* (a set of entities which own the same attributes and behaviours). Our system is best suited for homogeneous virtual environments (few families made of many instances).
- The *"stimuli"* convey interactions between entities.
- The *"virtual universe"* is the three dimensional environment where entities interact and behave.

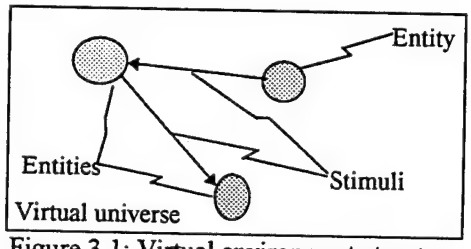


Figure 3.1: Virtual environment structure

A *"clone"* is a special type of entity which behaves as an interface between a user [Mouli93], an application [West92] [Carlsson93] or a robot and the virtual universe. An example of an application clone can be a modeller such as DeM²ons [Gaildrat93] a declarative multimodal modeller which is to be integrated with VIPER. The modeller entity will be able to provide some services to a user (in fact to a user's clone) in order to build a virtual world.

The purpose of this structure is to ease the definition of distribution schemes. Autonomous entities leads to a perfect encapsulation of the behaviour and state of an entity, and therefore ease distribution of entities: such an entity can execute its behaviour on a specific site, communicating with other entities through well defined stimuli.

In this part, in order to describe the structure of entities, we will present the inter-entity communication mechanism and the behavioural part of entities.

Entities and their environment interact in two main ways:

- from the environment to entities: a change in the environment produces a stimulus which is perceived by some entities. A stimulus is received by a specific sensor which interprets and then transmits information to the behavioural part of an entity.

- from each entity to the environment: once the information has been analysed, the behavioural part is able to change the internal state of the entity and if necessary commands actions to the effectors of the entity. Effectors act on the environment and while modifying it create new stimuli.

We propose to model these interactions with four elements: sensors, effectors, images and image spaces (Fig. 3.2). This model deals with environment internal interactions as well as those happening with the real world (e. g. for *"clone"* - user communication, devices such as datagloves and tracking systems are encapsulated in sensors).

To each type of stimuli, we associate a quadruplet: (image, image space, sensor, effector). An image is the perceptible part of an entity in the environment, in relation to a specific type of stimuli (e.g. a 3D shape is the perceptible part of an entity in relation to visual stimuli). An image space is mainly the set of images related to a certain type of stimuli. A sensor gets a "snapshot" of an image space which it filtrates and interprets. While executing an action an effector produces an image and then puts it into an image space. In fact, to each effector is first associated a unique image in the image space which is modified afterwards.

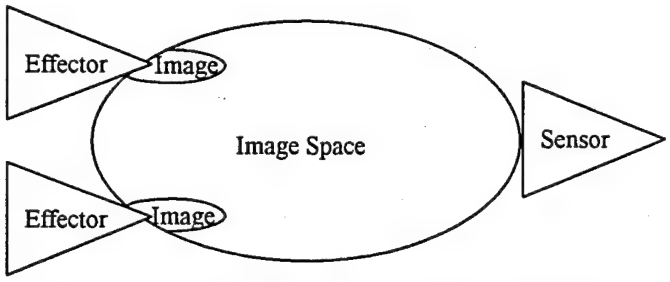


Figure 3.2: Inter-entity communication mechanism

Introducing an image space, which acts as a mediator in the inter-entity communication, is very useful in order to distribute a virtual environment (as will be explained in section 4.3).

The behavioural part of an entity consists in two components (Fig. 3.3): a *"reflex"* component, where entity reflexes to stimuli are defined, and a *"thinking"* component, where more complex behaviours can be defined. Behaviours can be implemented by finite state machines [Carlsson93], sensors/actuators networks [Wilhelms90] [Panne93], Prolog like rules [Rainjonneau90], intention generators [Xiaoyuan94], classifier systems [Torguet95] as well as interfaces to users, applications or robots.

The reflex component is made of a number of modules. Each module is assigned to a sensor in order to analyse its results and if necessary react, commanding actions to effectors and/or modifying its own state. The thinking component is triggered either by a clock or by any reflex module. This component defines the active side of the behaviour while the reflex component defines the reactive side.

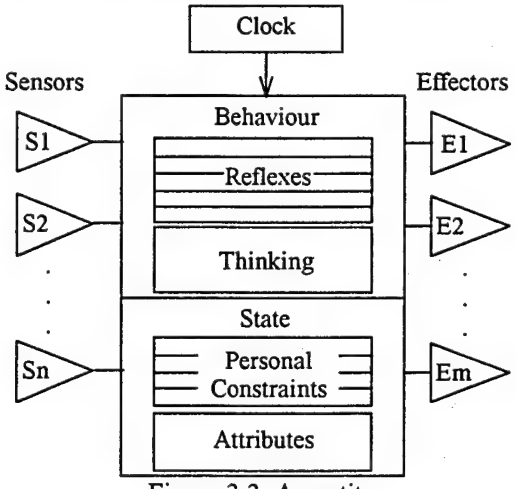


Figure 3.3: An entity

The entity has also its own state: a set of attributes (position, velocity, orientation, mass, 3D shape...) with a set of personal constraints (constraints which act on the entity attributes, e.g.: $velocity \leq maximum\_velocity$).

4. *Software architecture of VIPER*

The software architecture of VIPER consists in four layers (Fig. 4.1):

- At the bottom, the distributed platform, which is composed of a set of sites (workstations, multicomputers...), creates a virtual machine based on a message passing system like PVM [Sunderam90]. This layer encapsulates the communication system used by VIPER.
- Above this layer, an object-oriented concurrent programming environment encapsulates data distribution and an SPMD (single program multiple data) model [Moisan93].
- The third layer is the entity model, already described in the third section of this document.
- The topmost layer allows the definition of specific virtual worlds and the choice of their distribution scheme.

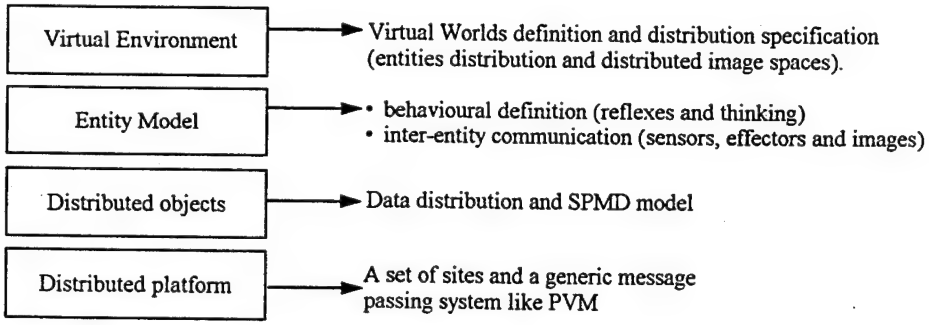


Figure 4.1: The software architecture of VIPER

4.1. The distributed platform

The distributed platform is the set of sites which take part in the simulation of the virtual environment. It is logically divided into four sets of resources: computing, rendering, storage and input/output resources. Each site is at least referenced into one of these sets. For example, a graphics workstation is referenced in all those sets.

Rendering resources are the set of sites which have render specific hardware. A graphics board and a three-dimensional sound rendering board are good examples of rendering resources. These resources are usually connected to sensorial devices such as head mounted displays or headphones. This set is also divided into subsets of similar rendering resources.

Input/output resources are the sites which have input/output ports used by VIPER. This set is divided into three groups: acquisition resources which allow access to virtual reality devices (such as datagloves and tracking systems), standard input/output resources (console) and operating resources which operate external systems (such has robot control boards). Acquisition and operating resources are further divided into subsets of similar devices.

4.2. Distributing an entity over the platform

In order to be easily distributed over the platform, each entity is divided into a number of parts called "sub-entities" (Fig. 4.2). These sub-entities are: rendering, input/output, behavioural and general sub-entities.

Each rendering sub-entity is composed of render specific sensors and attributes. Each input/output sub-entity is composed of acquisition sensors or operating effectors and attributes that are specific to a type of input/output. The storage sub-entity is made of all storage effectors and storage specific attributes. The behavioural sub-entity is composed of all the other sensors and effectors, reflexes,

thinking, personal constraints and behavioural specific attributes. The general sub-entity is made of the set of attributes which are not specific to one of the other sub-entities.

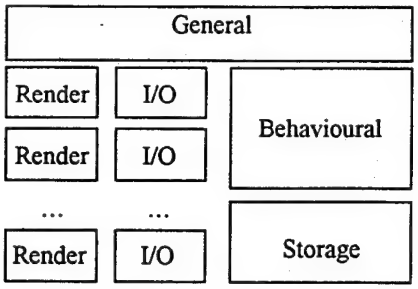


Figure 4.2: An entity divided into sub-entities

Each rendering sub-entity is given to one of the matching rendering resources. Each input/output sub-entity is similarly given to a matching resource. The behavioural sub-entity is given to one of the computing resources. The storage sub-entity is given to one of the storage resources. The general sub-entity is duplicated over all resources where the entity has a sub-entity. Practically, all entity resources ought to refer to the same site (e.g. a multiprocessor graphics workstation) or a few tightly connected sites. And the system always tries to get this property verified.

4.3. Distributing Virtual Worlds

This distribution of an entity introduces a functional parallelism which is internal to the entity. A more general parallelism within the virtual universe can be achieved. A virtual universe is a homogeneous environment inhabited by entities which behave in a similar way.

Object parallelism has been mainly used in existing systems because of its simplicity [Kazman93a] [Snowdon93]. The main advantage of this parallelism is to maintain a good load balancing. Obviously if each site is given the same load of entities each one would become equally loaded. Moreover, a strict encapsulation of entities allows the introduction of dynamic load balancing [Kazman93b].

However, in order to detect and manage direct or indirect interactions (collisions or information exchange in the environment), communication between sites introduces a great number of problems (bottle-necks on an interaction specific site for example).

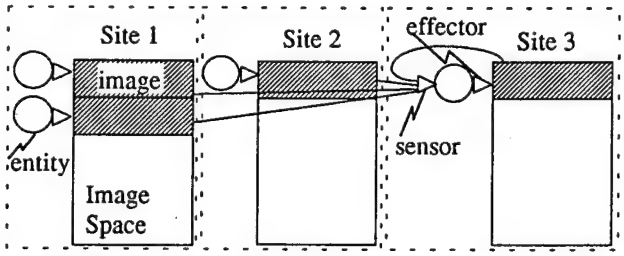


Figure 4.3: A fully distributed image space.

VIPER proposes to solve these problems with distributed image spaces. Communication between entities through image spaces has the advantage of being well defined and easily manageable.

We have currently defined three kind of distributed image spaces:

- fully distributed image spaces. In those image spaces, images are only situated on their entity computing resource and are remotely accessible. For example in Figure 4.3, when each entity effector creates an image of the entity for the first time, each image exist on the site where it has been created. And afterwards, modification of these images are only locally done. The entity on the third site is able to get every images present in the image space. This is done totally transparently: an access to a distant image is exactly the same as an access to a local image as far as the sensor is concerned. The only

difference being, obviously, access time. We think that this first type of image space is interesting when there are few sensor doted entities which retrieves information only from time to time, whereas effectors modify images very often. An example of such an image space can be a low frequency radar image space, where entities images are their position in space and sensors are radar which examine the image space with a low frequency.

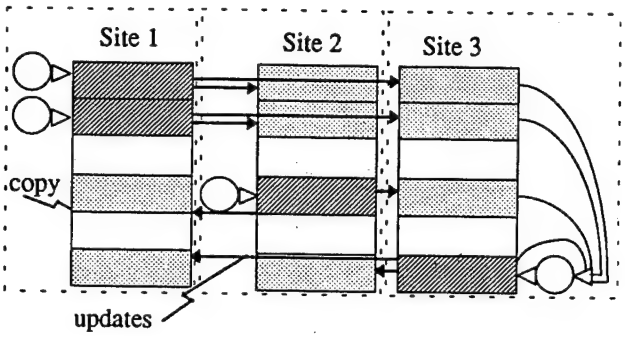


Figure 4.4: A fully duplicated image space

- fully duplicated image spaces, where images are duplicated over all computing resources and image updates are propagated. For example in Figure 4.4, when each effector creates an image, the image is created on every site the image space is accessible from. Each modification of an image is thereafter propagated to all sites via update messages transparently sent by the image space itself. When, for example, the sensor of the third site entity examines the image space, it only looks at local copies of each images. This type of image space is very interesting for image space where there is a lot of sensors sensing the image space often. An example of such an image space is the one used to mediate visual communication between entities. In this image space, entities outputs their changes in appearance (such as a position modification when moving) and other entities sense these changes in order, either to modify their behaviour or to display a view of the 3D world to their users (for user's clones entities).

- smart duplicated image spaces, where images are duplicated over computing resources which needs them (i.e. if there is, currently, a sensor connected to the image space) (Fig. 4.5). This is a derivative of duplicated image spaces, useful when there are few sensors (less than one at each site). An example of such an image space can be a sound mediating image space, which manages distribution of sounds produced by users or virtual objects and listened by users which have sound producing boards.

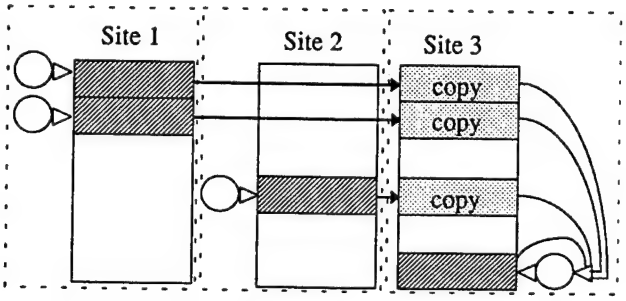


Figure 4.5: A smart duplicated image space.

However, in order to better solve some of the previously mentioned problems we think that we should take into account that, in most communication types, two entities may only interact if they are close enough. This can be done by associating topological information to image spaces.

A logical spatial partition of image spaces seems interesting for worlds which are logically divided into rooms linked by portals (doors, openings...). Each entity is associated to the room in which it is spatially present, and each site will only manage interactions which involve its entities (i.e. needs only the part of each image space which is local to its entities rooms). For example, if we consider a virtual building, an

entity may only interact with entities located in the same room or in adjacent rooms (if there are open doors).

A regular space partition can also be considered [Zyda93]. In this case, the virtual world space will be divided into grid squares of a constant size. Such a built grid can either be two dimensional (in case of applications in which altitude doesn't matter much: vehicle simulation for example) or three dimensional (applications where altitude is very important: flight simulation for example). Each entity is associated to the grid zone in which it is spatially present, and each site will only manage interactions which involve its entities (i.e. needs only local parts of each image space). An example of the use of topological information is presented in Figure 4.6. In this figure, the plane is only able to detect two vehicles because the image space, being aware of plane sensors limitations, only take into account a part of the virtual world when listening to vehicle images updates.

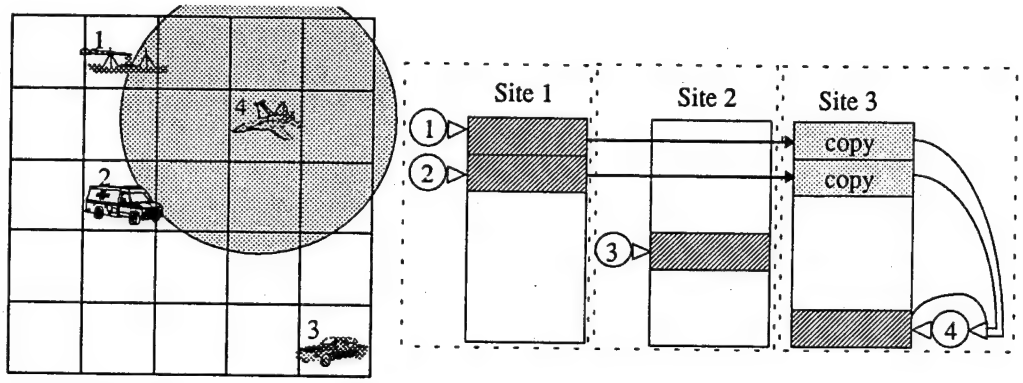


Figure 4.6: A smart duplicated image space with topological information.

An a priori choice for a distribution model is difficult to make. In fact this choice is really application dependent. For example, for a flight simulation a regular image space partition seems interesting, while a logical space partition seems more useful for an architectural walk-through. In VIPER the application developer is able to choose for each world and for each image space a specific distribution scheme. Moreover, the developer can specialize an already defined image space in order to better suit his needs.

Thus, the developer can quickly prototype his application using existing general distributed image spaces. And then, in order to increase efficiency, he may try other distributed image spaces or explicitly implement new optimizations algorithm. For example, the developper of a vehicle simulation application might choose a smart duplicated image space to mediate visual communication between entities (vehicles), then he might prefer a regularly partitionned image space, and eventually implement dead reckoning of vehicles.

5. *First results and Conclusion*

A first implementation of VIPER is being developed. It is based on PVM over an heterogeneous network of workstations (SGIs, HPs and SUNs). The choice of the PVM communication system, has been dictated in order to rapidly prototype VIPER. Indeed, PVM offers interesting tools like xpvm, which ease the definition of parallel applications and their tuning. However, the overhead added by this communication environment doesn't seem to match large scale virtual environments requirements. Thanks to the encapsulation of the communication system, VIPER can, nevertheless, be easily ported over any more efficient communication system.

VIPER is being developed using the C++ programming language which offers very interesting tools in order to develop generic constructs. C++ templates, for example, allows the definition of generic distributed image space which can be parameterized by a specific class of image (and eventually derived), in order to suit application needs.

The entity model, fully distributed image spaces and fully duplicated image spaces have already been implemented and are being benchmarked. Now we are developing our first application: a modeller which gives users the ability to create new shapes while deforming simple ones. This first application will allow us to estimate the efficiency of VIPER when confronted to a real problem.

This first implementation will then be extended with smart image spaces and virtual worlds topological information. The integration of a multicomputer in the platform is also being studied.

References

[Bricken93] W. Bricken, G. Coco, "The VEOS Project", Technical report, Human Interface Technology Lab, University of Washington, 1993.

[Carlsson93] C. Carlsson, O. Hagsand "DIVE - a Multi-User Virtual Reality System", Proceedings of VRAIS'93, Seattle, September 1993.

[Gaildrat93] V. Gaildrat, R. Caubet, F. Rubio "Declarative Scene Modelling with Dynamic Links and Decision Rules Distributed Among the Objects", IFIP International Conference on Computer Graphics ICCG93 Bombay, February 1993.

[Kazman93a] R. Kazman "Making WAVES: On the Design of Architectures for Low-end Distributed Virtual Environments", Proceedings of VRAIS'93, Seattle, September 1993.

[Kazman93b] R. Kazman "Load balancing and latency management in a distributed virtual world", 3rd International conference on Cyberspace, May 1993.

[Moisan93] B. Moisan, Y. Duthen, R. Caubet "Tools for SPMD object-oriented programming" EUROMICRO'93 Barcelona 6-10 September 1993.

[Mouli93] R. Mouli, Y. Duthen, R. Caubet "In VitrAm (In Vitro Animats, a behavioural simulation model)" 2nd IEEE International Workshop RO-MAN'93, Tokyo, 3-5 November 1993.

[Panne93] M. van de Panne, E. Fiume "Sensor-Actuator Networks" Computer Graphics SIGGRAPH'93 proceedings, pp 335-342, 1993.

[Rainjonneau90] S. Rainjonneau, Y. Duthen, R. Caubet "An Object Oriented Approach including an Inference Engine for Behavioural Simulation and Advanced Animation", TOOLS3 Sidney, pp 379-385, December 1990.

[Snowdon93] D. N. Snowdon, A. J. West, T. L. J. Howard "Towards the next generation of human-computer interface", Interface to real and virtual worlds, Montpellier France, pp 399-408, March 1993.

[Strommer93] W. M. Strommer, J. G. Neugebauer, T. Flaig, "Transputer-based virtual reality workstation as implemented for the example of industrial robot control", Interface to real and virtual worlds, Montpellier France, pp 137-146, March 1993.

[Sunderam90] V. Sunderam "PVM: A framework for Parallel Distributed Computing", Concurrency: Practice & Experience Vol. 2 No. 4, December 1990.

[Torguet95] P. Torguet, H. Luga, Y. Duthen, R. Caubet "A model for a user-adaptive response to interactions in Virtual Worlds", submitted to the Graphics Interface '95 conference, 1995.

[West92] A. J. West, T. L. J. Howard, R. J. Hubbold, A. D. Murta, D. N. Snowdon, D. A. Butler "AVIARY - A Generic Virtual Reality Interface for Real Applications", Virtual Reality Systems, sponsored by the British Computer Society, May 1992.

[Wilhelms90] J. Wilhelms, R. Skinner "A *Notion* for Interactive Behavioral Animation Control" IEEE Computer Graphics and Applications, May 1990, pp 14-22.

[Xiaoyuan94] T. Xiaoyuan, D. Terzopoulos "Artificial Fishes: Physics, Locomotion, Perception, Behavior" Computer Graphics SIGGRAPH'94 proceedings, 1994, pp 43-50.

[Zyda93] M. J. Zyda, D. R. Pratt, W. D. Osborne, J. G. Monahan "NPSNET: Real-time Collision Detection and Response" The journal of Visualization and Computer Animation, 4(1), 1993, pp 13-24.

Fine Object Manipulation in Virtual Environment

Ryugo KIJIMA
JSPS Research Fellow
Department of Mechano-Informatics,
Faculty of Engineering, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113, Japan
Phone: +81-03-3812-2111 (ext. 6369)
FAX: +81-03-3818-0835
E-mail: kijima@ihl.t.u-tokyo.ac.jp
http://manda.t.u-tokyo.ac.jp/
Michitaka HIROSE
Department of Mechano-Informatics,
Faculty of Engineering, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113, Japan

Abstract

We have developped two calculation method for the manipulation in virtual environments. Impetus Method is based on only one phenomena, collision between the fingertip and the object. We start with the simple manipulation to push the object with one degrees of freedom and expand it to treat with more complicated phenomena. RSPM realizes the detailed manipulation. By using this, the object can be grasped, manipulated by 3 fingertips.

1: Introduction

Natural and realistic object manipulations in virtual environments (VE) are important because the true application necessarily contains the object manipulation[8]. Moreover, the operator can recognize the law that dominates the behavior through the act of manipulation (active presence) in addition to the (passive) presence through the visual sense (Figure 1). Untill now, the realized manipulations have been generally based on gestures, and these manitulations seem to be very simple, symbolic, and not realistic.

The manipulation is realized by calculating the behavior of the object driven by a virtual hand. The behavior of the virtual object is an artifact. We can define any behavior. It appears that we can easily generate the behavior similar to that in the real world by calculations based on physical laws. There are, however, cases in which physical laws cannot be applied, especially in VE without force-feedback.

There have been many attempts and inventions for easier manipulation. Although some of them work well and effectively, there is a difficulty to combine different manipulation algorithms according to the various situations where the object locates in. What seems to be lacking is to explore the nature of difficulties in such calculations, and to systematize the methodology.

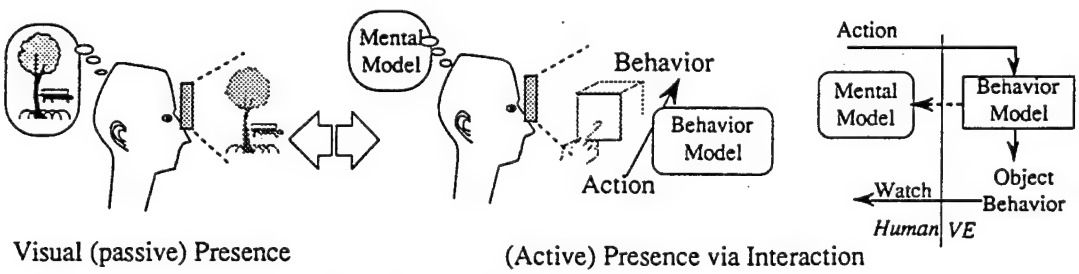


Figure 1 Realistic Manipulation Causes Active Presence

The goal of this study is to develop effective manipulation algorithms, and to make clear how we should manage different manipulation calculations. Although this paper does not fully acheive this goal, two example manipulation calculations with fingertips in VE without force feedback, are developed, and several discussions are shown based on this development to generalize the behavior calculation.

First, authors discuss the calculation for object manipulation generally.

Secondary, the "Impetus method" is described. It is based on a simple phenomenon, the collision between fingertip and object surface. We start with simple movements with 1 degree of freedom. This is naturally expanded to represent movement with 2/3 degrees of freedom, static and dynamic friction, stiffness, rotational moment, and restriction on a surface. This method belongs to the category of semi-dynamics.

Third, a "representative spherical plane method" for grasping and manipulating object by 3 fingertips, is described. It is based on the restricted interaction among three fingertips and a sphere that represent the object. This method enables detailed, fine manipulation by 3 fingertips. This belongs to the region of kinematics.

At last, the discussion of the above methods and future work is described.

2: Generation of Object Behavior

The calculation of the interaction or of the object's behavior in VE has different characteristics from those in popular Graphical Usr Interface (GUI) such as drag, pull-down, etc,. In the case of VE, it is more complicated. Generally, it has 3 degrees of freedom, and utilizes multiple contact point (fingertips, etc.). The number of states of the manipulated object are larger in VE than in GUI's. The largest difference is the detection to which model the behavior of the object belongs. In the case of VE, it should be based on geometrical information while those in GUI's are mainly based on symbol information such as a mouse click.

Symbol information simplifies the detection in exchange for giving up some detailed behavior. For example, in a VE, if the gesture of a hand (symbol information) mainly decides whether the hand grasps the object or not, the object is grasped without considering the geometrical relation between the fingertips and the surface of the object. These calculations based on symbol information are not suitable to generate detailed behavior.

2.1 Problem and Assumptions

In the case of VE without force feedback, the position/orientation of the hand or fingertip are monitored by sensor(s), and this data forms a virtual hand that interacts with the object. Namely, the hand's position as input is used to drives the object, and there is no output to the real hand. The data flow between the real hand and the virtual object is basically only one way.

In the case of VE with force feedback, most general method is, to sense the position of the hand, to calculate the interaction based on the position, and to display the impedance, force or the relation between position, velocity and force. Nevertheless, the data flow between the real hand and the virtual object is two way (Figure. 2.1).

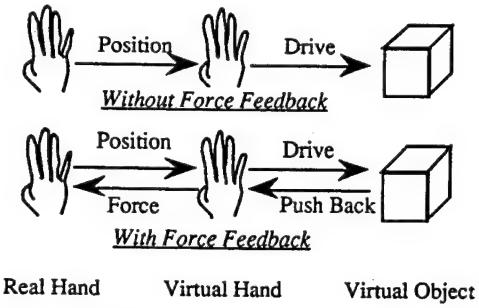


Figure 2.1 Dataflow

What is important here is: (Figure 2.2)

1: Only the position is obtained as input from the real world

This means we cannot include the term of force in the calculation. If we introduce some assumption that defines the relationship between the force and the situation (position, velocity and the other term that has been obtained), the force can be achieved. Because the assumption dominates the all of the calculations which are to follow, it should be chosen very carefully [6].

2: There is no output from the virtual world to the real hand

This means the movement of the real hand cannot be restricted, or cannot be modified by the reaction force from the virtual object.

Therefore, the position of the real hand could easily become different from that of the virtual hand. This is a serious problem because the calculation based on the position data implicitly assumes that it is based on the actual position of the real hand. To solve this problem, an assumption that defines the relationship between the real hand and virtual hand is needed. Also this assumption should be chosen very carefully [6].

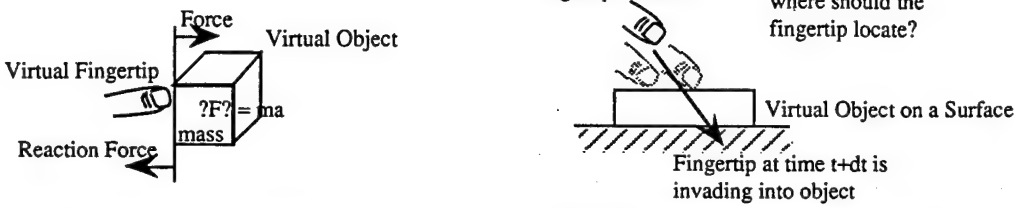


Figure 2.2 Calculation Problems in VE without Force Feedback

2.2 Kinematics, Dynamics

As an example, let us think about the case where a solid object is grasped by three fingers. This activity can be divided into three phases (Figure 2.3).

(a): Free

No finger comes into contact with the surface of the object. The behavior of the object can be calculated based on physical laws such as Newtonian Physics. If the world contains gravity, the object will fall freely. If the world contains viscous resistance, the velocity of the object will decrease gradually. Because the term of force appearing in the calculation is not the force between finger and object, there is no problem. This belongs to the category of dynamics.

(b): Restrictive

If the fingers grasp the object tightly, the position and orientation of the object have a strong relation to the positions of the fingertips. This can be calculated based on Kinematics. Kinematics as part of Mechanics, can deal with restrictive phenomena, such as, a pair of gears that engage each other, a set of arms that are connected with links, etc. The behavior of a grasped object is similar to such phenomena. Kinematics does not utilize the term of force. Therefore this belongs to the category of Kinematics-like methods. Section 4 corresponds to this region. Kinematics-like methods simplify the phenomena, avoiding the term of force, the need of wider bandwidth (higher calculation and sensing cycle)[5][7].

(c): Boundary

When some of the fingers are very close to the surface of the object, Kinematics is not suitable because the relation between the fingertips and the object will change rapidly. Also Dynamics as a part of Physics is not suitable because there is a force between the fingertip and the object, and because it suits for uniform or "flat" phenomena. Dynamics as part of Mechanics deals with the term of force explicitly, but it is focused on several mechanisms, introducing assumptions to simplify the theory and the calculation. Roughly saying, this category is similar to that of Dynamics. Section 3 corresponds to semi-Dynamics.

The calculation basing on symbol information such as the gesture of hand neglects this category. The symbol information divides the state of the object compulsorily into (a) and (b).

Thus with force feedback, the physical model for this category is not perfect. For object manipulations by robot-manipulator, there are several hypotheses and theories.

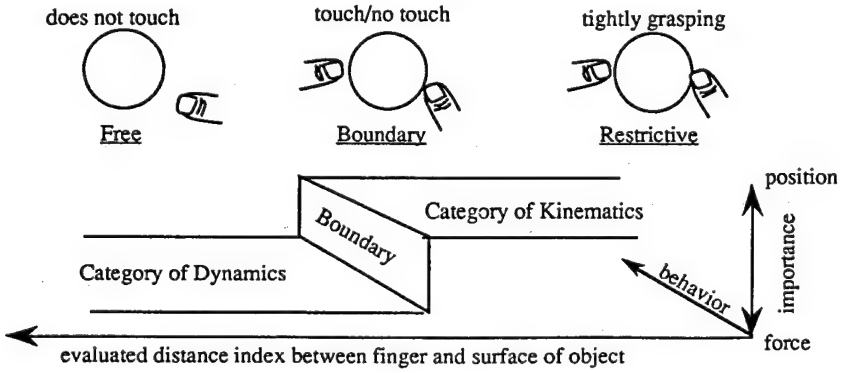


Figure 2.3 State of Object and Applicable Methods

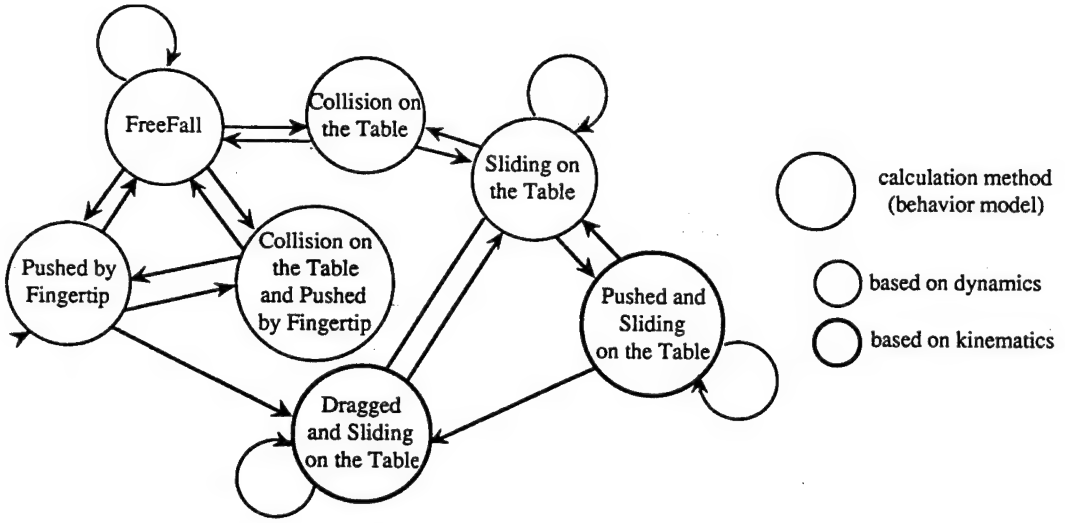


Figure 2.4 An Example of a Set of Behavioral Laws and State Transition

2.3 State Transition

The term "situation" can be defined as a set of parameters that describes both the state of the object and the state of the surrounding environment. For example, let us suppose that the virtual world consists of a virtual hand and a virtual object. The state of the object is a set of parameters such as the position of the object, the size, the velocity, etc. Because there is only the hand besides the object, the state of the surrounding environment is a set of parameters such as the position of the hand, gesture, etc,. The term means the combination of them. All the parameters form a configuration space that is used in computational kinematics [4]. One situation is a point in configuration space.

The object changes its behavior over these categories, and one category contains multiple regions of situation. One region corresponds to one calculation model of behavior.

When a calculation model is defined, the applicable region in the configuration space is restricted. Therefore the detection of which region the object belongs to, or which is a suitable calculation model for the object, is important to manage the over all calculations [3].

The fragmentary inventions of calculation that are not systematized, easily declined to error. The minor error in one model of behavior sometimes causes unreasonable state transition and changes the overall behavior largely.

3: Impetus Method

3.1 Basic Concept

This method aims to calculate the behavior of solid objects, manipulated with fingertip(s). Applicable manipulations and neighboring regions are pushing an object with finger(s) as shown in (Figure 3.1).

We start with defining clearly the set of axioms including the fundamental quantity that causes the movement of the object. The second step involuves detecting the applicable region from the axioms. The third step is to expand this to more complicated phenomena systematically.

In other words, the third step is the purpose, which is dependent on the second. The second step connects the state transitions and models correctly, which need to be clearly defined.

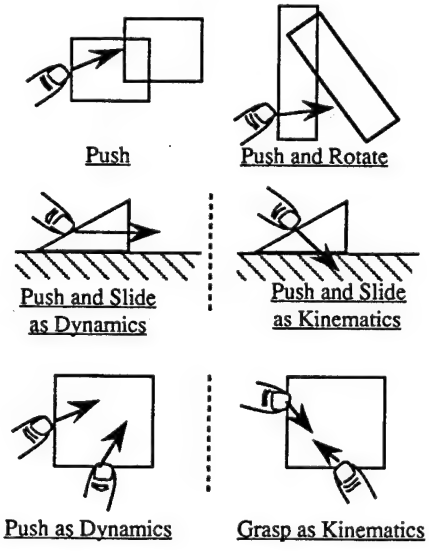


Figure 3.1 Applicable Phenomena and Neighboring Region of Impetus method

3.2 Axioms

The axioms are the following:

Law 1: "The reason for the change of motion is impulse

force, which is defined as the invasion vector of the fingertip"[1]. This needs the following assumptions to enable calculation:

Assumption 1: "Reaction force from the object to the finger is negligible". This requires several secondary assumptions such as the mass of object is zero, etc.

Assumption 2: "Any element cannot invade into the object"[2]. Not only at the time of the calculation point on the time axis, but also at the time between the calculation point, finger, other objects cannot invade into the inside of the object being manipulated.

3.3 The calculation and the detection of the boundary of applicable region

Figure 3.2 shows a basic calculation where one fingertip is pushing the surface of an object. Here, all the movements have only one degree of freedom.

At the time-index = n, the position of the fingertip is outside the object. At the next time segment of calculation, the fingertip position (gotten from sensor) invades into the object. Between time n and n+1, a collision occurrs. Using linear interpolation, the correct collision time and the trajectory of the fingertip and the object are calculated. Namely, the object position at n+1 is calculated from the fingertip position at n, n+1, object position and velocity at n, using simple interpolation.

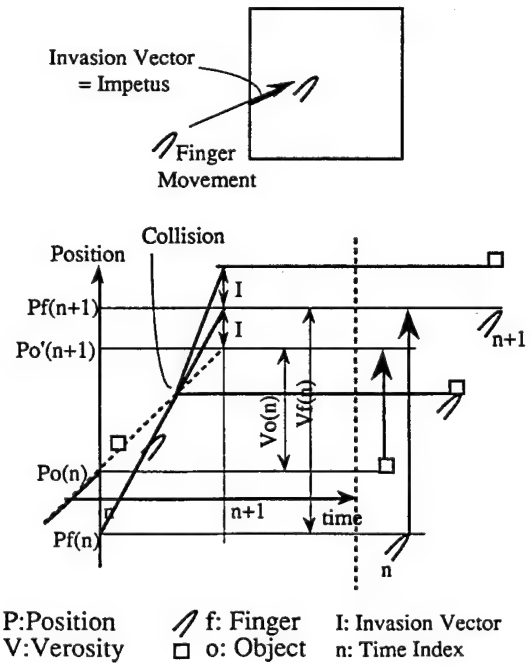


Figure 3.2 Calculation Based on Invasion Vector

$$Po'[n+1] = Po[n] + Vo[n] * dt$$

$$I = Po'[n+1] - Pf[n+1]$$

$$Po[n+1] = Po'[n+1] + I$$

As the invasion vector increases/decreases, the distance from fingertip to the surface of the object increases/decreases respectively. The invasion vector acts similarly as an impetus or impulse force.

The impulse from the fingertip is modified systematically by attributes such as collision factor, mass/rotational inertia ratio, dynamic friction coefficient, static friction limit, restriction on the surface, etc. It must be noticed that these (virtual) attributes are not the same attributes as in physics, but cause similar effects as those in Physics. Hence, we can control the properties of the phenomena easily.

(Collision factor)

The Impetus I in the equation (1) is modified by collision factor "e", which should range from 0.0 to 1.0.

$$Po[n+1] = Po'[n+1] + e * I$$

(Damping)

Next, a damping coefficient d is introduced. This represents phenomena such as the friction between objects being manipulated and their surrounding environment. As damping increase, the movement of object decreases. Impetus I in the equation (1) is modified in the following manner by introducing damping factor "d", which should range from 0.0 to 1.0.

$$Po[n+1] = Po'[n+1] + e * d * I;$$

(3D movement)

This is naturally expanded into movement with 3 degrees of freedom.

(Friction)

Static and the dynamic friction between the finger and the surface of the object are introduced. Until now, only the magnitude of the Impetus is modified by 2 factors. Here, the Impetus is divided into the normal element that is orthogonal to the object surface, and the parallel element that is parallel to

it.

Only the parallel element is modified to represent the dynamic friction. As the dynamic friction coefficient increases, the parallel element decreases. The dynamic friction coefficient should ranges from 0.0 to 1.0.

parallel element -> parallel element * dynamic friction coefficient

To detect whether or not a state belongs to the static friction region or dynamic friction region, the ratio of the parallel element/normal element is used. If the situation belongs to the static friction region, the parallel element is not modified.

The static friction limit should be larger than the dynamic friction coefficient.

if (parallel element/normal element > static friction limit) state = dynamic friction

else state = static friction

(Rotational Movement)

This is expanded into rotational movement by introducing a rotational Inertia ratio "r". The inertia ratio is similar to the ratio of rotational inertia by mass. This indicates the relative easiness of rotation to that of parallel movement. In the case that the mass of the object concentrates around the gravity center, it can easily be rotated compared with the ease of parallel movement. In the case that the mass exists mainly on the surface of object, it is relatively difficult to rotate.

Here, the Impetus is divided into two components. One is the parallel component that is parallel with a vector from the fingertip to the center of the object. Another is the rotational component that is the rest of the Impetus. This should cause the rotational movement.

rotational Impetus = rotational component * r * distance from the center of the object to the fingertip.

The rotational impetus is used to cause rotation. In our experimental system, this is directly converted to rotation for one time (from time n to n+1), continuous rotation is not supported.

(Restrictive movement on a plane)

The restriction of movement on a plane is introduced. This is achieved by simply eliminating the component that meets at right angle to the restriction plane.

3.3 Region detection

The applicable region of this method is detected by investigating whether or not the result of calculation satisfies assumption 2 or not. The problem is that "the result satisfies assumption 2" does not necessarily mean "the values used in the calculation satisfy assumption 2". In order to solve this problem, we introduce a restriction to the calculation as follows: "If the object satisfies assumption 2 in the region of time [t1.. t2), it satisfies the assumption 2 at t2."

3.4 Experiment System

Figure 3.3 shows the system for experiments. The computation of the behavior and the generation of graphics are performed on IRIS VGX-210 workstation. Polhemus IsoTrak and DataGlove is used to sense the fingertip point. All the program is written on VisAge that is an toolkit to generate virtual environment. VisAge is developed by the first author of this paper, and is consist of libraries and tools to develop the virtual reality application. It contains a simple management structure for behavior calculations. Objects are maintained in

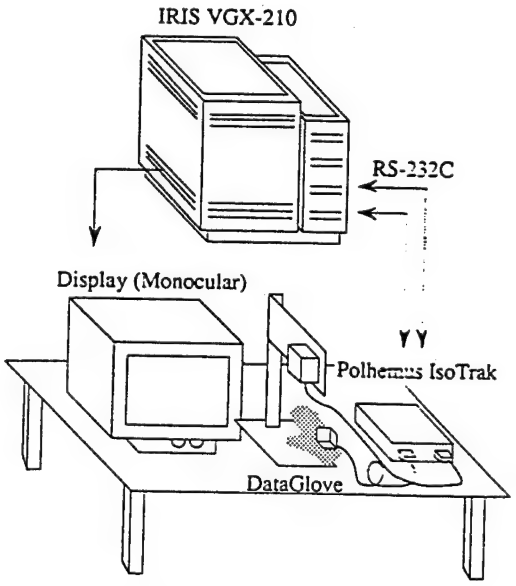


Figure 3.3 System for Experiment

a form of a tree. Each object is defined as a set of attribute and a pointer to a function that defines the behavior. The function is called automatically by the management structure referring the tree. As a programming style, it is recommended to divide the function into two parts, the detection of applicable region and the behavior calculation itself. The state transition is implemented by changing which function the pointer indicates.

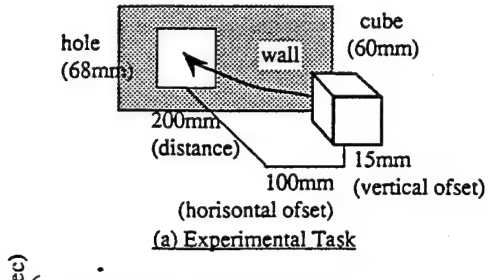
The experiment in section 4 is performed on the same system.

3.5: Experimental Result

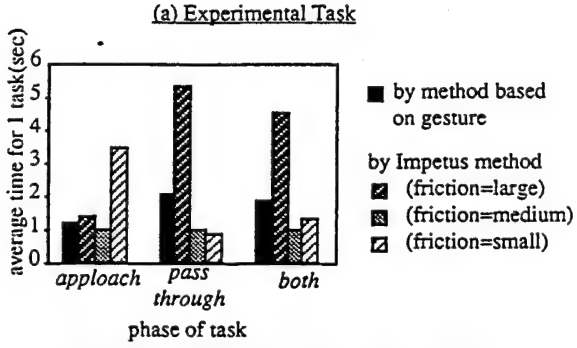
The experimental task was to place an object in the indicated region by pushing the object with one finger. Case 1 has 3 degrees of freedom parallel movement in free space. Case 2 has 2 degrees of freedom parallel movement and 1 degrees of freedom rotation on a plane. In both cases, the result shows that this manipulation is easier as compared to general symbolic manipulation.

Especially in case 1, the user used 2 states of friction properly according to the phase of task. This contributes to the improvement of performance. (Figure 3.3, 3.4)

Also the results showed that the user could "feel" the (virtual) attribute (friction ratio). In the real world, physical quantities are receipt as stimulus, cause the senses. Virtual attributes cause sense through the recognition of relations between the motion of the hand and the behavior of the object.



(a) Experimental Task



(b) Methods vs Task Time for Each Phase of Task

Figure 3.4 Experiment 1

4: Representative Spherical Plane Method

4.1 Basic Concept

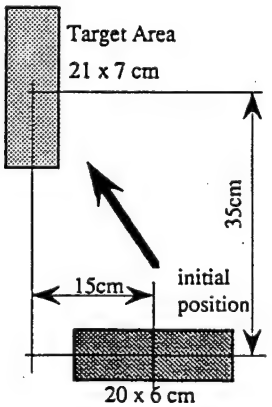
In this section, RSPM (Representative Spherical Plane Method) is detailed. This is for grasping and manipulating an object with 3 fingertips.

To begin with, let us introduce a sphere (RSP) with radius r that represents an object. The fingertips manipulate this sphere, and the object behaves similarly as this sphere. RSP moves according to the position of 3 fingertips, therefore, 3 fingertips only slide on the RSP while it is grasping the object. This movement of RSP is defined as the behavior of the object.

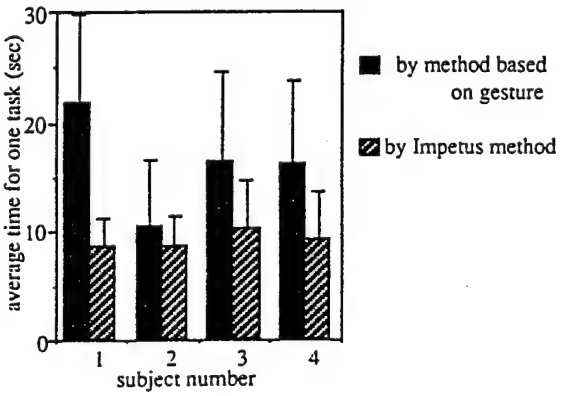
The merits of this method are as follows:

- Fine Manipulation with fingertips

The freedom of a finger is larger than that of the back of the hand, and a finger is more accurate. A finger is a "fine" part of the hand. In conventional gesture based manipulation meth-



(a) Experimental Task



(b) Average Time to Complete One Task

Figure 3.5 Experiment 2

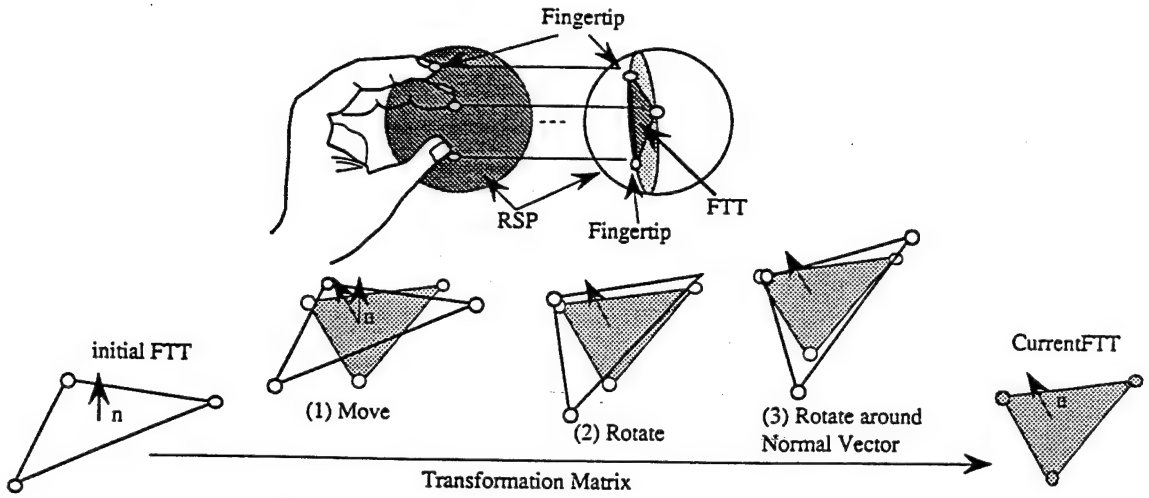


Figure 4.1 Representative Spherical Plane Method

ods, the position and orientation of grasped objects are gotten from the back of hand, and not from the finger. Fingers are utilized only for gesture detection. On the contrary, via RSPM, user can drive objects by their fingertips. This means the user could manipulate objects with more accuracy.

- Uniformity of Manipulation;

The user can manipulate any object with the same feel of manipulation.

- Uniformity of Calculation;

Any object can be manipulated based on the same algorithm.

RSPM begins with the condition that all 3 fingers are involved in the object, and that 3 fingers can grasp RSP. The second condition is checked by calculating the radius of circum circle of the triangle whose vertices are 3 fingertips. The second condition is equivalent to that the radius of circum circle is smaller than that of RSP.

4.2 Behavior Calculation

The behavior of RSPM is calculated as follows.

When fingertips grasp the object, an initial matrix that represents the position and orientation of RSP is calculated.

While grasping, a temporary matrix that represents RSP at that time is calculated. The transformation matrix from the initial matrix to the temporal matrix is easily calculated.

Let us focus on the fingertip triangle (FTT) that is comprised from 3 fingertips. When fingertips grasp the object, the initial FTT is stored in the memory. While grasping, the FTT is achievd each time. The transformation from the initial FTT to each FTT is equivalent to that from the object when it becomes grasped to the object at each time.

If FTT does not deform, the matrix that transforms the initial FTT to each FTT can be calculated easily. But in fact it deforms.

The matrix is restricted as combination of a unitary (rotation) matrix and a translation matrix. To avoid deformation, the following calculation is used;

(1): To move initial FTT such that the gravity center locates at that of each FTT.

(2): To rotate it around the gravity center such that the normal vector becomes equal to that of each FTT.

(3): To rotate around the normal vector such that the vertexes become closest to corresponding vertexes of each FTT. As the performance criterion, authors adopted the summing of absolute angle between the vectors that is from gravity center to vertex.

The result is calculated directly in analytical way, not in repetitive way. The system cycle was 60Hz, which include retrieval of data from sensor, calculation, graphics generation on IRIS VGX-210 workstation.

4.3 Experiment

A simple experiment was performed. A target cube and another cube were displayed. Each surface was painted in different colors. The task was to manipulate the cube so that whose orientation matches with that of the target cube. Two

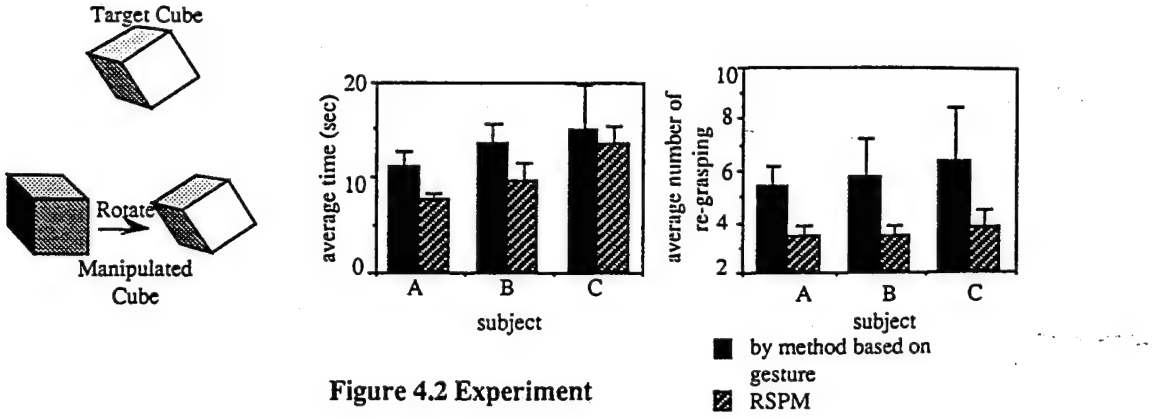


Figure 4.2 Experiment

methods was utilized. One was RSPM, another was general method based on gesture such that "if the gesture is FIST and the hand is near to the object then grasp" and that "the movement is the same as that of the back of hand".

The subjects performed 10 times per method. The average time to complete the task was measured. Users grasped object several times because the range of rotation was limited. They repeated grasping, rotating, releasing. Also the time of grasping was counted.

Figure 4.2 shows the result.

The average time for RSPM was shorter than that of method based on gesture. Also the number of re-grasp was smaller in the case of RSPM. This indicates the range of rotation increased in RSPM and RSPM was easier method than the conventional gesture based method.

5 Discussion and Future Work

5.1 Extending Impetus Method into Multi-finger Manipulation

Currently the Impetus Method deals the interaction between one fingertip and the object. The authors are testing an algorithm for the manipulation by two fingertips. Two impetuses from two fingertips are simply added and the result moves the object. This is partly generates the correct behavior. When one fingertip is invading into the object, the object is moved by one impetus. There is the case where the moved object contains another fingertip. This breaks the assumption 2 and causes the incorrect behavior. The authors are testing several algorithms to merge multiple impetuses and several detection algorithms. Although some of them work well in some cases, the suitable combination for any case has not been found.

Under some successful situations, the sequence as follows is achieved:

1: one finger continuously pushes the object.

2: two fingers pushed the object alternatively

3: two finger pushes the object at once

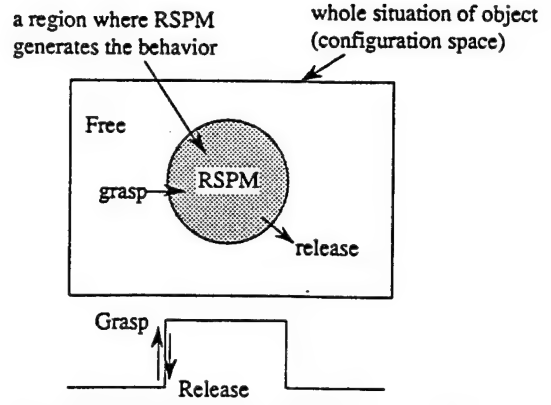
4: the object becomes not to satisfy the assumption 2, state is changed into the region of kinematics, the object is picked up by two fingers.

5.2 Discussion on Region Detection for RSPM

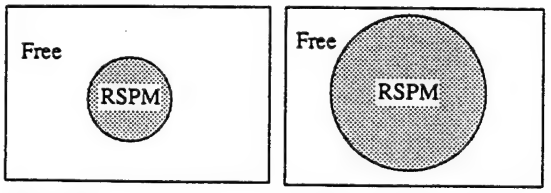
Although the experimental result shows the improvement of manipulation performance from the conventional manipulation method, RSPM leaves room for improvement. Much still remains to be done about the condition when RSPM begins (grasp object) and ends (release object), namely, the detection of the applicable region for RSPM.

We will begin with a simple case where only RSPM serves for manipulation calculation. In other words, the configuration space is divided into only 2 region, free and grasped (Figure 5.1(a)). The problem is that there are cases where the object is released while the user does not intend to release it, and the object is still grasped while the user intends to release it. The programmer can control the easiness to grasp and to release by the radius of RSP. As the radius increases, the region of RSPM widens, the object becomes easier to grasp and more difficult to release (Figure 5.1(c)). This parameter is not enough because it cannot increase/decrease both the easiness to grasp and release at once. To say as analogy, the control of the size of the region is not enough while it is useful to control the nature of manipulation.

To solve this problem, authors introduced the hysteresis (Figure 5.2). Before grasping (when the object is free), the radius of RSP is set to r1, and When the object is grasped, it is set to r2. When we give r1 a smaller value and r2 a larger



(a) Configuration Space is divided into 2 regions



(b) With Smaller Radius of RSP (difficult to grasp, easy to release)

(c) With Larger Radius of RSP (easy to grasp, difficult to release)

Figure 5.1 Applicable region of RSPM

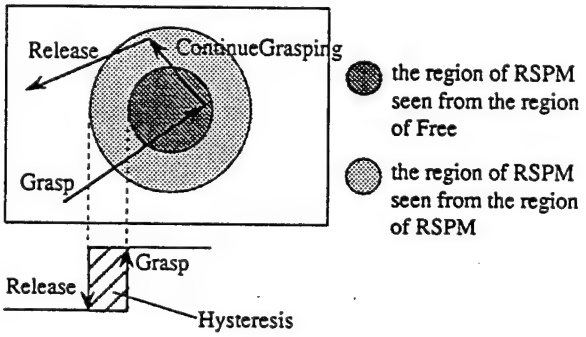


Figure 5.2 Introducing Hysteresis

value, it becomes more difficult to grasp and it becomes more difficult to release. This does not seem to improve the feel of manipulation. To say by analogy, we need to find the suitable shape of the boundary line of the region.

The discussion above is the region of RSPM itself. We may note briefly the further problem on the relation among different manipulation methods. Each method has an applicable region. As the discussion above shows, it is not easy to design the region solely. If there are several methods, it is needed to think about the overlapped region, about the void space of 2 regions.

6 Conclusion

First, the calculation for object manipulation was generalized.

Authors pointed out that calculation of object behavior was defined as a set of pairs of the calculation model and the applicable region, the region detection. The type of region was classified into 3 categories, Dynamics, Kinematics and the boudary category of them.

Second, Impetus Method as Dynamics-like method was described for the behavior calculation of the boundary category. It was based on a simple phenomenon, the collision between fingertip and object surface. Experimental result showed superiority to the conventional method.

Third, RSPM for grasping and manipulating object by 3 finger tips, was described as Kinematics-like method. Experimental result showed superiority to the conventional method

At last, the expansion of Impetus Method into the manipulation by multiple fingertips was discussed. Also the applicable region of RSPM method was discussed.

7 References

[1] David Baraff, "Fast Contact Force Computation for Nonpenetrating Rigid Bodies", Procs of SigGraph '94, pp.23-34, ACM (1994)

[2] David Baraff and Andrew Witkin. "Dynamic Simulation of Non-penetrating Flexible Bodies", Procs of SigGraph '92, pp.303-308, ACM (1992)

[3] Michael Gleicher. "Integrating Contraints and Direct Manipulation", Procs of Symposium on Interactive 3D Graphics '92, pp.171-174, ACM (1992)

[4] "Qualitative Reasoning", Section 8: Computational Kinematics, Toyoaki Nishida, Asakura Books (1993)

[5] Peter Schroder and David Zeltzer. "The Virtual Elector Set: Dynamic Simulation with Linear Recursive Constraint Propagation", Procs of Symposium on Interactive 3D Graphics '90, pp.23-31, ACM (1990)

[6] Jeffrey A. Thingvold and Elaine Cohen, "Physical Modeling with B-spline Surface for Interactive Design and Animation", Procs of Symposium on Interactive 3D Graphics '90, pp.129-137, ACM (1990)

[7] Andrew Wilkin, Michael Gleicher and William Welch, "Interactive Dynamics", Procs of Symposium on Interactive 3D Graphics '90, pp.11-21, ACM (1990)

[8] David Zeltzer, "Autonomy, Interaction, and Presence", PRESENCE, Vol.1, No. 1, pp.128-132, MIT Press (1992)

Consistent Grasping Interactions with Virtual Actors Based on the Multi-sensor Hand Model

Serge Rezzonico[1], Ronan Boulic[1], Zhiyong Huang[1]
Nadia Magnenat Thalmann[2], Daniel Thalmann[1]

1) LIG - Computer Graphics Lab, Swiss Federal Institute of Technologies
Lausanne, CH-1015 Switzerland
fax: +41-21-693-5328
email: {rezzoni; boulic; huang; thalmann}@lig.di.epfl.ch

2) MIRALab-CUI, University of Geneva, 24 rue du Général Dufour
Geneva, CH -1211 Switzerland
fax: +41-22-320-2927
email: thalmann@cui.unige.ch

Abstract

This paper proposes a general framework to enhance grasping interactions of an operator wearing a digital glove. We focus on a consistent interpretation of the posture information acquired with the glove in order to reflect the grasp of virtual artifacts. This allows manipulations requiring a higher skill in virtual environment and also improve interactions with virtual human models. A handshake case-study highlights the application range of this methodology.

key words : grasping, virtual human, digital glove

1 Introduction

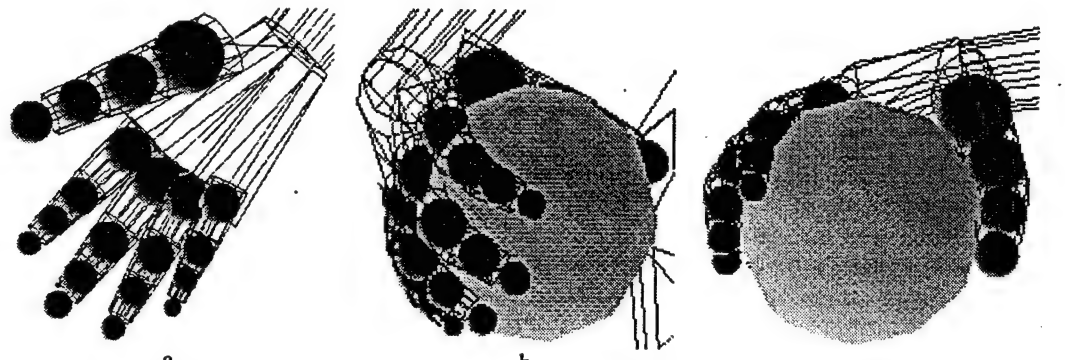
With the advents of synthetic actors in computer animation, study of human grasping has become a key issue in this field. The common used method is a knowledge based approach for grasp selection and motion planning [RG91]. It can be completed with a proximity sensor model [EB85] or a sensor-actuator model [vdPF93] for the precise position control of the fingers around the object [MT94]. This method results in an automatic grasping procedure. Moreover, due to the 3D interactive tools widely available today, we decided to study interactive grasping of virtual objects while wearing a digital glove device. Such an approach is also interesting for Virtual Reality where more elaborated hand interaction is now possible with recent generation of digital glove. way In this context, we map the real posture of the digital glove on a sensor-based virtual hand in order to ensure a consistent collision-free grasping of virtual objects and to provide a consistent visual feedback. In a second stage, this process can drive the grasp behavior of a virtual human model with a classical inverse kinematic control applied to the arm, or a larger fraction of the body [PB90]. More elaborated control approaches of the arm have been proposed but this is beyond the scope of this paper (see [L93], [HBMTT95]). Both automatic and interactive grasping are integrated within the TRACK system [HBMTT95] hence allowing such grasping interaction as a handshake of an operator with a virtual human model.

We first recall the principle of the multiple virtual sensor grasping prior to develop the consistent virtual grasp with the digital glove. The framework of interactively driving the

grasp behavior of a virtual human model is then outlined and a case study is presented. A discussion summarizes the performances, the interest and the limitations of the current state of the system. A short section presents the implementation details prior to the general conclusion.

2 Automatic Grasping with Multiple Virtual Sensors

This section briefly recalls the interest of automatic grasping based on virtual sensors (without digital glove). Our approach is adapted from the use of proximity sensors in Robotics [EB85] and the sensor-actuator networks [vdPF93]. More precisely, we use multiple spherical sensors for the evaluation of both touch and distance characteristics as proposed in [MT94]. They were found very efficient for synthetic actor grasping problem. Basically, a set of sensors is attached to the articulated figure. Each sphere sensor is fitted to its associated joint shape, with different radii. The touch property of any sensor is activated whenever colliding with other sensors or objects (except the hand components). This is especially easy to compute with spherical sensors (Figure 1).



a b c

Figure 1. The virtual hand with sphere sensors (a); while grasping a sphere (b, c)

The sensor configuration is important in our method because, when the touch property of a sensor is activated in a finger, only the proximal joints are locked while distal joints can still move. In such a way, all the fingers are finally wrapped around the object, as shown in Figure 1b,c. When grasping a free form surface object, the sphere sensors are detecting collision with the object. We do not discuss more on collision detection which is beyond the scope of this paper (see [MT94], [K93]).

The automatic grasping methodology is the following [MT94]: first a strategy is selected according to the type and the size of the object to grasp. A target location and orientation is determined for the hand frame and it is realized with the well known inverse kinematics approach. The next stage is to close the fingers according to the selected strategy (e.g. pinch, wrap, lateral, etc.) while sensor-object and sensor-sensor collisions are detected. Any touch detection locks the joints on the proximal side of the associated sensor. The grasping is completed when all the joints are locked or reaching their upper or lower limit.

3 Interactive Grasping with a Digital Glove

When an operator is wearing a digital glove the joint values acquired with the device are normally used to update the visualization of the hand posture. Such approach is pertinent as long as the device is used to specify commands through posture recognition [SZF89]. Among these commands there usually is a symbolic grasp of virtual objects where the relative position and orientation of the object is maintained fixed in the hand coordinate system as long as the grasp posture is recognized. Such approach is suitable for pick-and-place tasks of rigid objects and it is not our purpose to change it.

However, as the virtual environment is becoming more and more complex, especially with the advent of virtual humans, hand-based interactions also evolve in complexity. The limitation of the current approach mostly comes from the rough relative positioning of the hand and the object which does not convey a clear understanding of the action to perform with the object. As everybody has experienced him/herself, we adopt different grasping postures of a same object according to the function we intend to exert with that object because different degrees of mobility are involved for these different tasks (e.g. giving or using a screwdriver). Moreover, the immersion and the interaction in a virtual environment may not only reduce to a matter of pick and place but also imply abilities requiring a greater skill. In such a case the hand associated with the digital glove device provides a high dimensional space not only as a posture space (for command recognition) but also as a goal-oriented space (for precise manipulation or modification of virtual objects with additional tools). For example, interaction with non-manufactured deformable entities (articulated or continuously deformable objects) is best performed with direct hand interaction as it is our most elaborated tool to translate our design intention into action. in such context we need to evaluate precisely their mutual contact location.

3.1 Interactive Grasping Automata

Within this extended application context of the digital glove, it becomes crucial to display a posture of the hand consistent with the on-going manipulation of the virtual object. For this reason, we propose now a new approach for the interactive and consistent grasping of virtual entities with the interactive grasping automata (Figure 2).

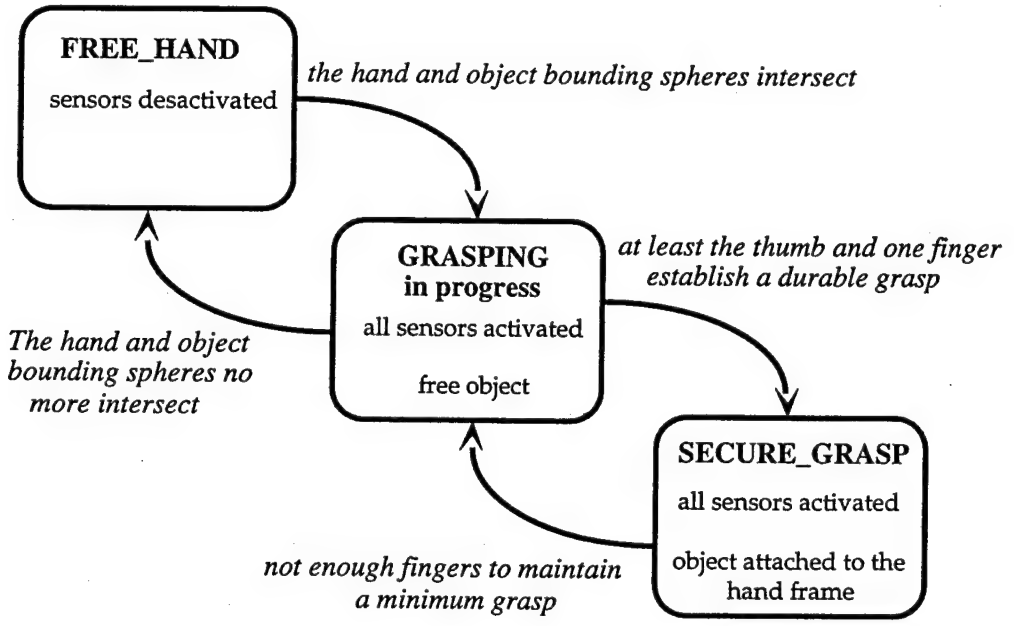


Figure 2 : The interactive grasping automata

In our method we consider three different states of interactive grasping :

FREE HAND : the hand is freely moving in space without holding any object. The hand posture is displayed as measured with the device. Whenever the hand bounding sphere intersects the object bounding sphere, we enter in the "GRASPING in progress" state.

GRASPING in progress : the touch property of the sensors is continuously evaluated to adjust the posture of colliding fingers with the object to grasp (***the object***

is fixed or moving in a world coordinate system). Whenever the hand bounding sphere no more intersects the object bounding sphere, we return to the "FREE_HAND" state. On the other hand, if our simplified grasp condition is established, i.e. at least the thumb and one finger are maintaining a durable contact with the object, we enter the "SECURE_GRASP" state.

SECURE GRASP : the touch property of the sensors is still used to continuously adjust the posture of colliding fingers with the grasped object (***the object position is fixed in the hand coordinate system***). As soon as the simplified grasp condition vanishes, we return to the "GRASPING in progress" state.

3.2 Hand Posture Correction

Unlike the automatic grasping procedure, the interactive grasping procedure adjusts the hand posture by opening it rather than closing it. Even with the recent generation of digital glove it is difficult to adjust the grasp precisely so that the fingers establish a permanent contact without penetrating into the virtual object. This is due to the fact that we only a have a visual feedback without any force or touch feedback. However, it would be misleading to conclude that the automatic grasping procedure should also apply in this context. Basically, interactive grasping implies to ensure the highest autonomy of the operator and to provide means of correction rather than removing degrees of freedom.

It is more comfortable for the operator to freely move and close the hand according to the visual feedback of the virtual environment. So our working hypothesis is to rely on the operator to permanently close the grasping fingers slightly more than theoretically necessary. In such a way, the opening correction approach establishes a durable contact which overcomes the unavoidable small variations of hand posture and position (Fig. 3). An optional mode of *Assisted Folding* is also provided to guide the operator in searching the proper grasp posture. In this mode, any sensor initially situated between the first colliding sensor and the finger tip is brought to be tangent to the object. If the sensor is intersecting the object then the associated joint is opened otherwise it is closed. In such a way, the distal part of a colliding finger consistently wraps around the object (Figure 4). The correction algorithm is characterized by an opening-wrapping adjustment loop with eventual Assisted Folding for each colliding finger. So, for each time step, we have :

```
For each colliding finger

      For each sensor distal to the colliding one closest to the base
            (from base side to tip side of the finger)

            If the sensor is currently colliding

                  Unfold the closest proximal joint (wrist side)
                     until the sensor is tangent to the object
                     or the joint reaches its limit

            Else

                  If in Assisted Folding Mode

                        Fold the closest proximal joint (wrist side)
                           until the sensor is tangent to the object
                           or the joint reaches its limit
                  EndIf
            EndIf
      EndFor
EndFor
```

Figure 3 details all the stages of the opening-wrapping algorithm for one colliding finger with an elliptic shape (in 2D for clarity). In the example, the joints are all successively

opened because the distal sensors (on the finger tip side) are still colliding even after the correction of the proximal ones (on the wrist side). The algorithm begins by unfolding the finger base joint to release the first colliding sensor (fig. 3a,b). Then it unfolds the next joint to remove the following sensor (Fig 3b,c) and the same occurs for the last joint (Fig. 3c,d). In this case the final finger posture consistently wraps around the object.

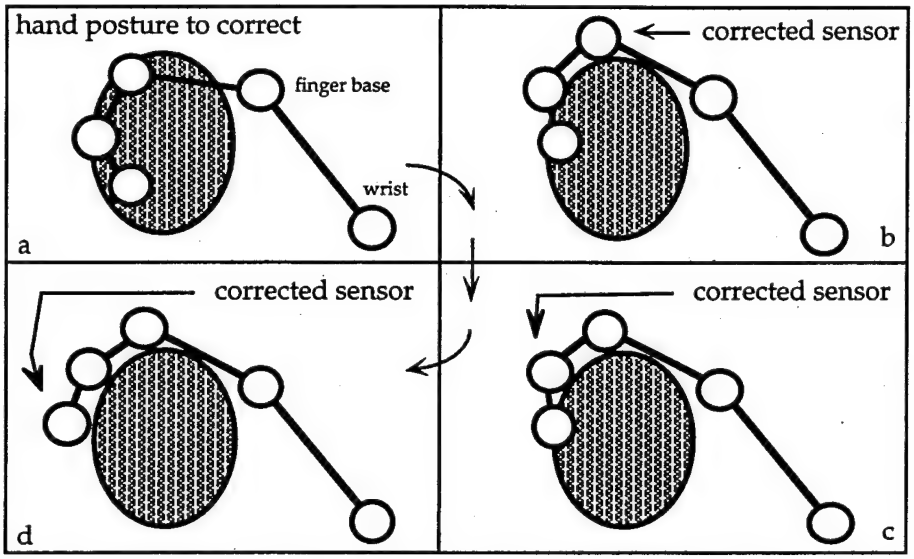


Figure 3 : an example of the opening-wrapping adjustment loop for interactive grasping

The Assisted Folding mode is especially interesting whenever the tip side of the finger is not in the operator field of view (Figure 4). This happens for a large class of grasping postures and objects to grasp. In such a way the operator is given a hint about the full grasp posture of the colliding fingers. Then, from that continuous visual feedback he/she can adjust the hand position, orientation and posture in order to perform a desired grasp. In figure 4 example, the first corrected sensor relies on opening the associated joint while the next two distal sensors are brought to be tangent to the rectangle shape by closing their associated joint.

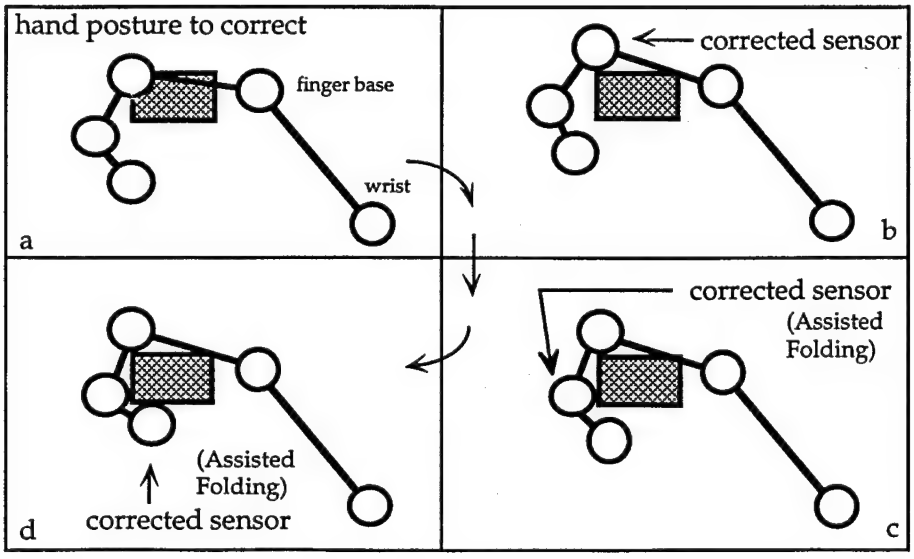


Figure 4 : an example of the opening-wrapping adjustment with assisted folding

4 Integration within the TRACK system

The TRACK animation system is dedicated to human animation design [BHMTT94]. More recently we have begun to evaluate the potential of virtual environments through 3D interaction with virtual humans. As such, both grasping approaches, automatic and interactive, are key features to integrate within the TRACK system. We now present the current state of this integration and outline the progressive intermixing of both techniques in order to allow complex grasping interaction.

- Automatic grasping of static volumic primitives and polygonal surfaces has been defined for both one and two hands grasping for the virtual human (Figure 1) [HBMTT95], [MT94]. So it is already possible to exchange handshake between virtual humans by activating their hand motion and automatic grasp in sequence rather than simultaneously.(figure 5).

- Autonomous interactive grasping is already performed on volumic primitives of the virtual environment and is to be extended on polygonal surface.

- Guiding one virtual human's hand with a 6D device as the Spaceball or a digital glove is an alternate approach to the knowledge-based selection of the grasp posture and positioning relatively to the object. The Automatic grasping closure is then performed to establish a wrapping grasp based on the sensor collision detection while closing the fingers. In such a context, the device must stay in the reachable area of the virtual human's hand otherwise the virtual human has to be globally displaced.

- Finally, the most complete intermixing of both techniques is to fully map the digital glove position and posture on the virtual human's hand and to manage it according to the interactive grasping approach. In such a way the operator can fully handle the grasping function of one virtual human model and interact with other virtual humans. The other virtual humans can in turn respond to the operator according to the automatic grasping approach as long as the grasping target is not moving. For moving objects as in a handshake context, the operator's virtual hand is the target of the virtual human's hand and both have to use the interactive grasp with assisted folding. In such a way the virtual human closes its fingers only when colliding the operator's hand. Moreover this requirement overcomes the operator's hand postural changes and simultaneous position variations.

Figure 5.: handshake between virtual actors

5 Results

Two interactive grasping experiments are presented here. They deal with the grasping of regular volumic primitives as a sphere and a rectangular volume with various sizes. First figure 6 exhibits the hand model (as provided with the digital glove library from Virtual Technologies). We limit the size of the virtual objects to grasp in order to permit single hand grasping. The following figures shows simultaneous view of the real hand posture as acquired with the digital glove and the one displayed as the result of the interactive grasping approach. The sensors are displayed as cubes to reduce the amount of polygons.

Figure 6 : hand model provided with the digital glove of Virtual Technologies

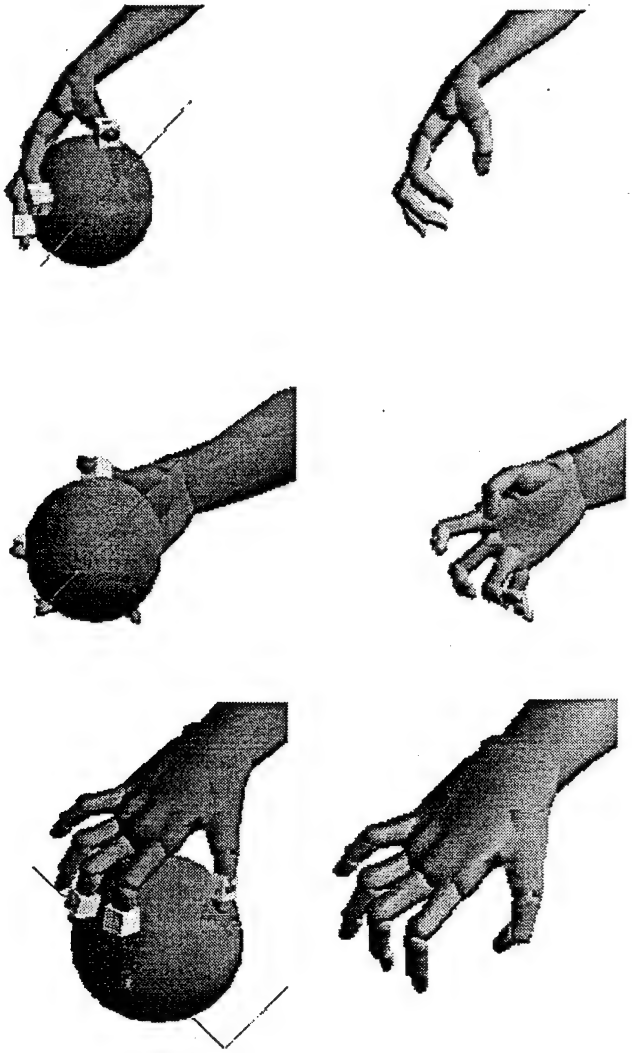


Figure 7 : the corrected (left) and the real (right) hand posture for various viewing angles and object sizes (sphere primitives)

6 Discussion

The interactive grasping experiments were realized at interactive rate of approximately 10 images per second on a workstation screen with the management of up to 20 spherical sensors and the display of the high resolution hand (2600 polygons), the object and the sensors. Although a small delay was perceptible between performance and display of the corrected hand posture, is was not a decisive aspect.

Basically, interactive grasping proved to be a valuable tool for the interactive design of realistic grasping posture for human animation system. The interactive refresh rate is the main criteria for that purpose and our experiments have demonstrates this capability. So, from the visual feedback of the corrected grasp, an operator is able to achieve a good perception-action control loop in order to continuously adjust the grasp into a realistic posture. In our default setting the display is completed with color coding information as described now. Each state of the interactive grasping automata is associated with a different color of the whole hand : pale skin color for FREE_HAND, blue for GRASPING_in_progress and green for SECURE_GRASP. Moreover, whenever a sphere sensor is colliding, its color changes from dark gray to light gray. This additional symbolic information significantly enhances the operator feedback.

Regarding Virtual Environments applications the realism of the grasping posture may not be an essential factor in favor of our approach. In such a context it is often not the point to behave exactly as in the real world. A VE operator is more concerned with a greater manipulation skill rather than a more realistic posture. In that aspect our approach also improves the manipulation of objects by comparison with the well known symbolic grasp. In the symbolic context, the operator has to perform well defined hand postures (at least two distinct ones) so that the recognition system separates properly the grasp from the release commands (not to mention that the object should be selected first). So, whenever the operator wishes to precisely modify the relative position and orientation of the object with respect to the hand (this is the proper definition of a manipulation), he/she has to grasp it, to reorient the hand-object system, release the reoriented object and move the hand freely to a new relative orientation. In our approach the reorientation is still performed in that way but the grasp is established in a much simpler way, just by touching the object with the thumb and another finger. Such procedure is much easier and faster that the one based on posture recognition.

As mentioned just before the finger manipulation skill is limited to set the begin and the end of the grasp. Modifying the relative orientation of the object with respect to the hand coordinate system is managed in the same way as symbolic grasping (see above). This procedure is tractable as long as we perform a light grasp involving a small number of fingers. Otherwise, the operator has to repeatedly close and open the hand which can be rapidly uncomfortable.

7 Implementation

We can either use the DataGlove from VPL or the Cyber Glove from Virtual Technologies. This latter has been retained for the performance evaluation all along the present experiments. The hand polygonal model provided by Virtual Technologies comes from the 3-D Dataset of Viewpoint DataLabs. The position and orientation of the Cyber Glove was acquired with one bird sensor from the "Flock of bird" device of Ascension Technologies. The interactive grasping was computed on a Silicon Graphics Indigo II Extreme. The virtual human and interactive grasping software are written in C language.

8 Conclusion

We have studied interactive grasping of virtual objects to improve the goal-oriented interactions of an operator wearing a digital glove device. The interactive grasping approach with the opening-wrapping algorithm ensures a consistent collision-free

grasping of virtual objects which proves to be a valuable visual feedback for the operator hence allowing to manage tasks involving a higher skill than before. Encouraging performances on a standard graphic workstation open the way for integration in fully immersive systems.

Interesting applications of this techniques appear as virtual human models also begin to invade virtual environments or as the digital glove begin to invest animation systems dedicated to human animation design. For example, our approach can drive the grasp behavior of a virtual human model in order to simplify all the grasping studies for production. Both automatic and interactive grasping are integrated within our TRACK animation system.

9 Acknowledgments

We wish to thank our colleague Tom Molet for the implementation of the virtual human model in the framework of the ESPRIT project HUMANOID, and Ramon Mas who has developed the first version of the automatic grasping approach. The research was supported by the Swiss National Science Research Foundation and the Federal Office for Education and Science.

10 References

[BHMTT94] Boulic R., Huang Z., Magnenat-Thalmann N., Thalmann D. (1994) Goal-Oriented Design and Correction of Articulated Figure Motion with the TRACK System , Computer. & Graphics, Vol. 18, No. 4, pp. 443-452.

[EB85] Espiau B., Boulic R. (1985) Collision avoidance for redundant robots with proximity sensors, Proc. of Third International Symposium of Robotics Research, Gouvieux, October.

[HBMTT95] Huang Z., Boulic R., Magnenat-Thalmann N., Thalmann D "A Multi-sensor Approach for Grasping and 3D Interaction" , Proc. of CGI 95, Leeds

[K93] Kamat V.V. (1993) A Survey of Techniques for Simulation of Dynamic Collision Detection and Response, Computers & Graphics, Vol 17(4)

[L93] Lee P.L.Y. (1993) Modeling Articulated Figure Motion with Physically- and Physiologically-based Constraints, Ph.D. Dissertation in Mechanical Engineering and Applied Mechanics, University of Pennsylvania.

[MT94] Mas R., Thalmann D. (1994) A Hand Control and Automatic Grasping System for Synthetic Actors, Proceedings of Eurographic'94, pp.167-178.

[PB90] Philips C. B., Zhao J., Badler N. I. (1990) Interactive Real-Time Articulated Figure Manipulation Using Multiple Kinematic Constraints, Computer Graphics 24 (2), pp.245-250.

[RG91] Rijpkema H and Girard M. (1991) Computer animation of knowledge-based human grasping, Proceedings of Siggraph'91, pp.339-348.

[SZF89] Sturman D.J., Zeltzer D., Feiner S.(1989) Hands-on Interaction with Virtual Environments, Proc. of ACM SIGGRAPH Symposium on User Interface Software and Technologies, pp 19-24

[vdPF93] van de Panne M., Fiume E. (1993) Sensor-Actuator Network, Computer Graphics, Annual Conference Series, 1993, pp.335-342.

Interaction Models, Reference, and Interactivity in Speech Interfaces to Virtual Environments

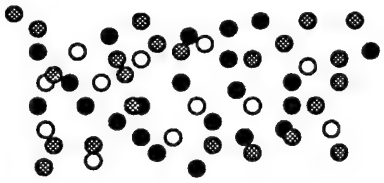
Jussi Karlgren, Ivan Bretan, Niklas Frost, Lars Jonsson
Swedish Institute of Computer Science
Box 1263, S - 164 28 KISTA, Stockholm, Sweden
diverse@sics.se

Abstract

The enhancement of a virtual reality environment with a speech interface is described. Some areas where the virtual reality environment benefits from the spoken modality are identified as well as some where the interpretation of natural language utterances benefits from being situated in a highly structured environment. The issue of interaction metaphors for this configuration of interface modalities is investigated.

Introduction

Virtual reality interfaces sometimes seem to be thought of as embodying a return to a natural way of interaction – the way we interact with the real world[1]. The interaction metaphors already introduced for VR (with some trimming and tuning and the addition of proper tactile feedback...), would then be sufficient for interaction. No learning would be required, as opposed to traditional interfaces – the natural interaction mechanisms are all there. This is a familiar mistake: it has been made repeatedly in the natural language-processing community. Not until recent years has it been widely acknowledged that conventions from other human activities do not always carry over directly to interactions with computer systems. We will give some examples to show similar oversimplifications regarding virtual reality technology.



"Select the grey marbles."

Figure 1: Just point and click.

The Naming Of Things Is A Serious Matter

"This" and "that" used deictically are physical world concepts easily defined and formalized for virtual reality interfaces in the form of direct manipulation mechanisms. However, they constrain their users to the here and now, even if "here" and "now" may be defined differently than in the physical reality. Human languages are by design a step beyond "this", "that", "here", and "now". They allow the user to refer to entities other than concrete objects, using set conventions: abstract concepts ("reality"), actions ("eating"), objects that are not here ("the dog

[1] "[We are] on our own again, after the long mediation of top-down authored experience (...)": Brenda Laurel, WIRED 1.6

Pim"), objects that are not present now ("last month's salary"), objects that cannot exist ("perpetuum mobile"), and objects selected for a property ("slow things"). In general, rendering the domain of interaction in terms of physical objects is not always appropriate – many things are difficult to portray[2].

"Where is the paper about virtual reality I sent to CHI last fall?"

Figure 2: Try this with gestures.

Virtual metaphors are conventions

The virtual world does not need to obey th laws of the physical: in the real world, lan guage is a means to change the world, an in a virtual world the world will be easie to change. Take something as simple as virtual table. Unlike its physical relative, i can change to accommodate the preference of the user. Similarly, the virtual world ca be instructed to transport us to somewher in the virtual space. Naturally, metaphors a virtual saw, a virtual pot of paint, a fly ing carpet, superpowers – to do this wit could be introduced, but they will not b more natural or less conventionally boun than use of language would be, on the con trary.

"Paint the table red and make it round."

"Take me to the moon."

Figure 3: Manipulating the world with language.

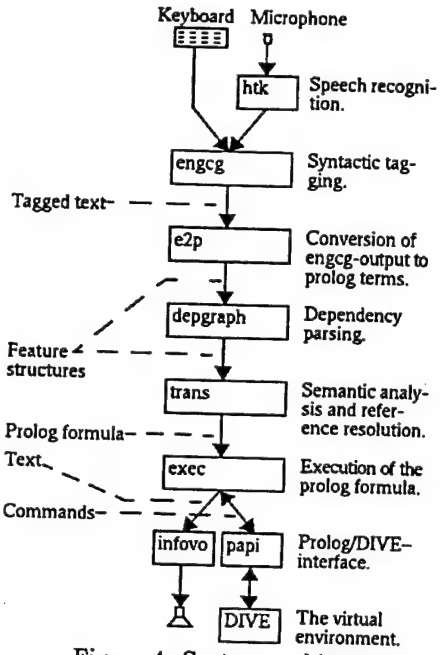


Figure 4: System architecture.

[2]This is the point of playing charades.

System sketch

Our system – DIVERSE (DIVE Real time Speech Enhancement) – is a speech interface to a generic virtual environment based on DIVE (Distributed Interactive Virtual Environment) that can be used with complex worlds modelled in a variety of formats (Carlsson and Hagsand, 1993). DIVERSE allows a user to select and manipulate objects in the world and move about in it. DIVERSE is implemented as a cascaded sequence of components. Speech recognition is done by means of a Hidden Markov Model system – HTK – which has been trained for the domain (Woodland et al, 1994). Text processing is performed by a general-purpose surface syntactic processor – ENGCG – which identifies syntactic roles and dependencies in the text (Karlsson, 1990). A resulting dependency graph is translated to a logical representation, which in turn is inspected for references to entities and objects and matched to the set of conceivable and possible actions. The resulting queries or commands are then sent to DIVE which manipulates or queries the world accordingly.

Interaction Metaphor

There is no obvious counterpart to the user for dialog with a system in a speech controlled virtual environment. There are several conceivable interaction models:

The basic metaphor of virtual environments is that of **Personal Presence**: the user is embodied in the real world through an actor or entity in it. This model poses problems for speech interaction – who will the user address? ("I now want to paint the house red...") This metaphor can be extended to that of **Proxy**, where users in effect ride on the back of a virtual entity. Users share the perspective, and can address and control their proxies at will "Sindbad: paint the house red!". An alternative similar to that of the proxy are the closely related metaphors of **Divinity**, where users give commands *as* a god to no obviously present counterpart but instead to the world itself: "Paint the house red!" or even "Let the house be red!"; or that of **Prayer** where users address commands in a similar fashion *to* a god.

Another extension of the basic metaphor of personal presence is that of **Telekinesis** where the objects and entities of the world themselves can be counterparts and interlocutors to users: "House, open your door!". Drawbacks include (1) the ability of an object or set of objects to participate in a dialog is far from obvious; (2) talking to objects not yet in the world will not be natural: "Three small red cubes, create yourselves!"; and (3) the need for object independent communication "Take me home". Of course, the last types of message could be addressed to some type of meta-object: a creation object or transportation object – in any case, the counterpart would be highly convention-bound.

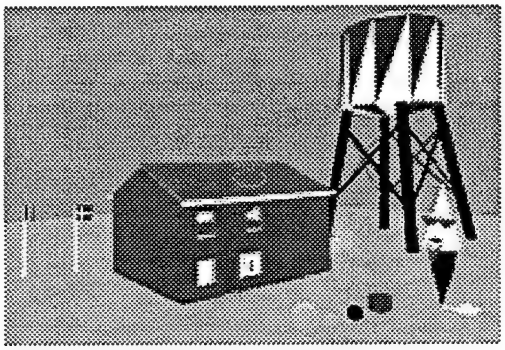


Figure 5: Interface snapshot with agent to the right.

A different type of interaction metaphor is that of an **Agent**. The agent model is different from other models in that it requires a separately rendered autonomous entity with communicative capabilities. The users will find a virtual, visually present, assistant or

agent to interact with. This is necessary to be able to integrate visual and spoken feedback naturally; with no feedback or interlocutor, the interaction situation would most likely be very unfamiliar and difficult to make use of. This is the interaction model we have chosen for our implementation of DIVERSE. A consequence of machine use of a single interlocutor is that the system's linguistic competence can be modelled in this agent through its visual characteristics, its gestures, its language, and so on – this will encourage convergence in one direction. Accordingly, the DIVERSE agent has been provided with a simple vocabulary and a small set of gestures.

Reference resolution – pragmatics

One of the most challenging problems of language understanding is that of reference resolution: of tracking what referents referential expressions refer to.

We are not even sure of what the characteristics of referents are: we have reasonable evidence from text studies that referring expressions in the text do not refer directly to other expressions in the text itself, but to referents outside it (e.g. Brown & Yule, 1983); similarly we have reasonable evidence that referring expressions do not refer directly to the "world", "knowledge base" or whatever we posit be the "reality" that the discourse is "about", but to some intermediate level, which we in the following will call *discourse referents*. We will make no claims about the characteristics of such referents: in our implementation, with the exceedingly simple task and object structure, we have yet had no need to implement an intermediate level. Our operations apply directly to the world.

Resolving which discourse referent a speaker or writer refers to is non-trivial: usually there are several possible candidates. In the general case, knowledge of the domain in addition to syntactic information and access to the discourse and other aspects of the situation that the language use occurrs in are usually necessary. Brown and Yule, e.g., (1983) mention several approaches involving multiple knowledge sources; an implementation by LuperFoy (1991) lists nine different sources her algorithms utilize, including Recency, Global Focus, various grammatical and lexical features, and some knowledge oriented features.

The knowledge sources used in the various approaches can roughly be categorized into two types: 1) situation specific features: recency, focus, and formal features of the referring expression; and 2) encyclopædic features, involving different kinds of world knowledge.

In DIVERSE we only have partial encyclopædic information. We have full knowledge of what objects exist in the world, and we have a certain hierarchical organization of objects with subparts, but there is no representation of object relations, roles, and world characteristics. What we do have is an excellent representation for discourse tracking, to analyze focus.

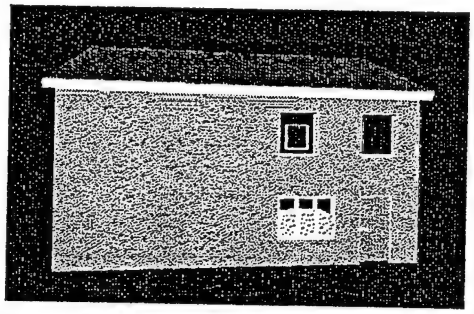


Figure 6: "Paint the house black." – What does "the house" refer to?

To concretize, the problem we need to solve is that of resolving what the referring expression "the house" in the user utterance

"Paint the house black." refers to as in figure 6, what the referring expression "it" refers to in the user utterance "Move it to the left." as in figure 7, and what the referring expression "a cube" refers to in the user utterance "Take me to a cube." as in figure 8.

The attentional state of the system is easily modeled by visual focus and the highlighting mechanism of DIVE; this means that where a pure text based system might have to deliberate about different candidate cubes – given that there are several in the world – in the example interaction in figure 7 a multimodal system will have a less vague situation with the pictorial accompaniment shown in figure 8.

```
Move me close to a cube.
```

Figure 7: What does "a cube" refer to?

We give each object in the world a focus grade, based on recent mention, highlightedness, gestural manipulation by the user, and above all, visual awareness. Definite noun phrases and pronouns are handled similarly, for now: the set of candidate referents is constrained by focus grade, and the candidate with the highest focus grade is chosen as a referent; for indefinite noun phrases the entire world is allowed as candidate set.

So, primarily, if an object is in the perceptual focus in the virtual environment, i.e. the agent has a high degree of *awareness* of it (Benford and Fahlén, 1993; Benford et al, 1994) it is a prime candidate for reference.

One of the actions available to users is to *select* or *point* at an object. An object which the user points at gets a high focus grade, with a rapid rate of focus decline after the pointing gesture has been completed. The command "Select *object*!" or even just "*Object*!" highlights the object. This is intended to be a method for users to pick out referents before issuing commands that process them. In addition, objects can also be highlighted as a consequence of a previous command.

Thirdly, we keep track of which objects have been referred to recently. If an object is in the textual discourse focus, i. e. in the recent *dialog history* it is a strong candidate for reference. An important design issue is how the dialog history is represented. To encourage users to refer to previously mentioned or manipulated objects, the discourse history can be made explicit: presumably the representations of likely candidates for reference will influence the actual references made. This must be studied empirically, with various varieties of DIVERSE implementations being compared to one another.

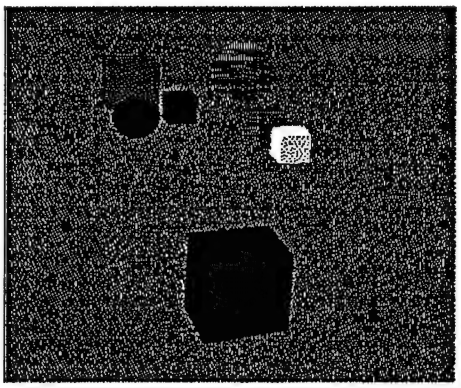


Figure 8: Now, what does "a cube" refer to?

The evidence from gestures, awareness status, previous commands, and discourse history is added together to determine which object is the one most likely to have been referred to. We have not yet determined exactly what the respective weighing of these methods will be: this is partly an empirical question.

Typical problems for text based reference studies are that the prototypical case, where a definite noun phrase refers to previously introduced referents and indefinites introduce new referents, is not that frequent (Fraurud, 1990). Thus, any algorithm for finding a referent for a definite noun phrase

will need a fair amount of world knowledge to pick a contextual sponsor or anchor for the referential expression. We have found that the visual awareness factor overrides the importance of most other channels, so that in an interaction, objects can be introduced as salient just by looking at them. If the user moves to look at a tree, and then says "Move the tree to the left." it is clear which tree is meant. And, if the visual awareness is given priority over other sources, the feedback given the users will always give users information of what is going on.

A typical view of the drawbacks of natural language as an interface tool, be it keyboard entered or spoken, compared with direct manipulation is given by Cohen (1992): "... another disadvantage [of natural language input] is that reference resolution algorithms do not always supply the correct answer in part because systems have underdeveloped knowledge bases, and in part because the system has little access to the discourse situation the user finds himself in, *even if the system's prior utterances and graphical presentations have created that discourse situation.* ... These ... world knowledge limitations undermine the search for referents of anaphoric expressions and provide another reason that natural language systems are usually designed to confirm their interpretations."

Bos *et al* have implemented EDWARD, a text and direct manipulation operating system for workstations (1994). They note that users sometimes lose track of selected objects: "we found ... users not always being aware of the state of the model world: the markedness of objects selected a while ago was sometimes forgotten or overlooked." In DIVERSE we may be able to expect slightly better user attention – visual awareness is much better determined; the view is fixed in EDWARD, whereas the user can change the view in DIVERSE, and as the visual focus overrides selection and highlighting of objects, a DIVERSE user can be expected to be more aware of the state of the model world and markedness of objects. Whatever the case may be on that count, Bos *et al* note that the mistakes the system makes do not seem to faze users; the errors are interactive enough for the user to accept them. Thus they partly answer Cohen's objections: in a highly interactive environment, errors do not matter; at least if the interface is honest about its abilities and cooperative as to displaying them. In our design, feedback is not a matter of asking the user for confirmation, but a view of system actions.

Errors do not matter

The interactive design of the DIVERSE interface is related to recent trends in natural language interface research, where the underlying problem of interactive interfaces, especially natural language interfaces, today is identified as that of a low degree of interactivity or "one-shot"-interaction, where users believe – regardless of system competence – that systems expect them to pose queries in one go (Bretan and Karlgren, 1993).

The conversational competence users expect from computers is extremely simple, which has been shown in a number of studies of natural language interfaces. This is specifically true for discourse structure, which has been shown to be modellable by an exceedingly simple dialog grammar, by examining the discourse structure of material obtained in Wizard of Oz simulation studies (Dahlbäck, Jönsson, and Ahrenberg, 1993). This can be explained by a fundamental *asymmetry of beliefs* between user and system (Joshi, 1982). Users do not expect computer systems to take responsibility for the coherence of a discourse, but expect to take full responsibility for the discourse management themselves. This is in contrast with naturally occurring dialog which is not only interactive but also *incremental*, i.e. in a form where both parties cooperatively build up referents and references during the course of a discourse.

To change this, the system must somehow display and make explicit what information it has for the user to refer to, and what assumptions about user intentions it makes; at the current point of sophistication, a high degree of interactivity and added communication channels to the system is arguably a better tool for raising system usefulness than adding functionality or intelligence to the existing channel, be it text, speech, or a rule based system (Chandrasekar and Ramani, 1989; Lemaire and Moore, 1994; Karlgren et al, 1994).

As indicated in the previous section, in DIVERSE we make use of the errors-do-not-matter principle to the extent that we will not worry about the system misinterpreting the occasional user utterance: as long as the interface is interactive we do not expect misinterpretations to bee too crucial a problem. More important than error handling is a broad acceptance of user utterances: every utterance should produce some effect.

The representation of the utterance is matched to representations of possible actions in the domain. If no good match is found, any referents that have been identified in the utterance are highlighted anyway, to facilitate users to continue the discourse, rather than starting from square one again. This is similar to recent ideas about how to generally design a natural language interface, using "non-threatening error messages that reiterate vocabulary and phrases the processor understands." (Zoltan-Ford, 1991).

Conclusions

Language is not only about conveying information[3]: it is a tool for acting in the world. Without immediacy with respect to the world it is used in, it is not natural language. Conversely, VR interaction without language does not take place in a natural or intuitive world. We are working on overcoming some of the most fundamental weaknesses of these two areas of interactive system design – through merging them.

[3] In fact, as an experiment, the reader is invited to approximate how large a percentage of language use the reader personally uses for conveying information.

References

Benford, Steve, John Bowers, Lennart Fahlén, and Chris Greenhalgh. 1994. "Managing Mutual Awareness in Collaborative Virtual Environments" *Proceedings of VRST'94*, Singapore. New York: ACM.

Benford, Steve, John Bowers, Lennart Fahlén, and Chris Greenhalgh. 1995. "User Embodiment in Collaborative Virtual Environments" *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'95)*, Boston. New York:ACM.

Benford, Steve and Lennart Fahlén. 1993. "A Spatial Model of Interaction in Large Virtual Environments" *Proceedings of 3d ECSCW* Milan: Kluwer.

Bos, Edwin, Carla Huls, and Wim Claassen. 1994. "EDWARD: full integration of language and action in a multimodal user interface" *International Journal of Human-Computer Studies*, 40:473-495.

Bretan, Ivan and Jussi Karlgren. 1993. "Synergy Effects In Natural Language-Based Multimodal Interfaces" *Proceedings of 1993 ERCIM Workshop on Multimodal Human-Computer Interaction*, Nancy:INRIA. (also available as *SICS Research Report R94:04*.

Bretan, Ivan and Jussi Karlgren. 1994. "Worlds without Words", *Proceedings of ERCIM Workshop on VR*, Stockholm:SICS.

Brown, Gillian and George Yule. 1983. *Discourse Analysis*. Cambridge: Cambridge University Press.

Chandrasekar, R. and S. Ramani. 1989. "Interactive communication of sentential structure and content: an alternative approach to man-machine communication", *International Journal of Man-Machine Studies* 30:121-148.

Cohen, Philip. 1992. "The Role of Natural Language in a Multimodal Interface", *In Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, Monterey: ACM.

Carlsson, Christer and Olof Hagsand. 1993. "DIVE, a Platform for Multi-User Virtual Environments", *Computers & Graphics*, 17:6.

Dahlbäck, Nils, Arne Jönsson, and Lars Ahrenberg. 1993. "Wizard-of-Oz Studies — Why and How", *Proceedings of the 1993 International Workshop on Intelligent User Interfaces*, Orlando:ACM.

Fahlén, Lennart, Charles G. Brown, Olov Ståhl, Christer Carlsson. 1993. "A Space Based Model for User Interaction in Shared Synthetic Environments" *Proceedings of the ACM Conference on Human Factors in Computing Systems (InterCHI'93)* Amsterdam:ACM.

Fraurud, Kari. 1990. "Definiteness and the Processing of NP's in Natural Discourse." *Journal of Semantics* 7:395-433.

Joshi, Aravind. 1982. "Mutual Beliefs in Question-Answering Systems", in N. V. Smith (ed), *Mutual Knowledge*, London:Academic Press.

Karlgren, Jussi, Kristina Höök, Ann Lantz, Jacob Palme, and Daniel Pargman. 1994. "The Glass Box User Model for Information Filtering", *Proceedings of the 4th International Conference on User Modeling* Cape Cod:ACM. (A longer version available as SICS Technical Report T94:09).

Karlsson, Fred. 1990. "Constraint Grammar for Parsing Running Text". *Papers presented to the Thirteenth International Conference On Computational Linguistics (COLING -90)*, H. Karlgren (ed.), Helsinki:University of Helsinki.

Lemaire, Benoît and Johanna Moore. 1994. "An Improved Interface for Tutorial Dialogues: Browsing a Visual Dialogue History". *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'94)*, Boston:ACM.

LuperFoy, Susann. 1991. *Discourse PEGS: A Computational Analysis of Context-Dependent Referring Expressions.* Ph D Dissertation. Austin:University of Texas at Austin.

Woodland, P.C., J.J. Odell, V. Valtchev and S.J. Young. 1994. "Large Vocabulary Continuous Speech Recognition Using HTK". *Proceedings of ICASSP'94*, Adelaide.

Zoltan-Ford, Elizabeth. 1991. "How to get people to say and type what computers can understand". *International Journal of Man-Machine Studies* 34:527-547.

The Metaphor : A Modern Holy Grail
Towards a methodology of choice

Olivier DUPUY,
Université de Marne la Vallée, Institut Gaspard Monge,
Centre d'Informatique Graphique, 2 rue de la butte verte,
93166 NOISY LE GRAND CEDEX
dupuy@univ-mlv.fr

ABSTRACT

Any software, be it graphic or textual, proposes an interface to browse a computer generated world and to give commands. More than the interface, metaphors of representation, navigation and action are judged by the user. With them, he will forge his knowledge of the goal and the usage of the system. The aspects and characteristics of these metaphors are fundamental for the adequation between the mental model and the task to be achieve. Virtual reality and mostly the presence quality amplify this requirement. In the face of these facts, this paper presents a methodology to find and combine the right metaphors.

Keywords : metaphor, mental model, virtual reality, representation, navigation, interaction techniques, human factors

INTRODUCTION

Using metaphors is not a computer centred activity but a human thought practice [8]. Everyday language abounds in illustrating speech, *painting* a talk with *figures* which help understanding. Politicians form *sides* when individuals aim *arrows* or *blows*. *Sportsmen struggle* for *victory* and so on. An oratory *defeat* by lack of arguments is unpleasant, like a real one. Parables, allegories and others have their place in our vision of the world. They bind a tie between rational knowledge and interpretation.

Permanently, people build internal representations of themselves and of the objects they interact with, these objects being material or not, animate or inanimate things, sentient or not. These *mental models* explain and predict the interactions. Their lack or slow growth is a factor that limits independence of action in everyday life or in any other domain. But with time, practice and knowledge, these models become more and more abstract and so more effective. The naive users « physical » representation of a computer as a set of memory, processor and disk may for example evolve to a more potent model with file systems, informations flows, process...

Concerning interfaces, most of software packages present a visual metaphor explaining what to do and how to do. It should be a direct extension of the thought, the first step towards the elaboration of a mental model, presenting the state of the system, suggesting means of movement or action. The literal definition of a *metaphor* is the use of a linguistic category and its attributes instead of another one. In computer pidgin, it means to disguise the internal state of specific data or of the computer with a visual translation more common and more suited to navigate, manipulate or what is necessary, it is a world with a set of coherent rules.

Some actions can be done inside the world of the metaphor while others require a dialogue with the *metaworld*, an expression regrouping all actions or things not internal to the metaphor. This aspect is often neglected and sometimes inconsistent with the

remainder of the metaphor but is requisite. A good example of a inner and outer side of an interface is given by the *vi* editor where we sometimes leave the typewriter mode to save a file. A metaphor given by the handbooks and illustrated by the interface is the link between the mental model and the data. It should be a guide to design an internal representation and a tool to traduce the will of interpretation, navigation or modification.

Using metaphors is a common practice for teachers but to be efficient, we privilege in our choice ones with likeness of objects by their structural relations more than by their physical properties, a natural approach but least efficient. A metaphor of towns (computers) tied by roads (networks) is often used by engineers to a explain to a non-specialist a need for a bigger flow of cars (data). In fact, we do not need the two domains being related, a certain distance can even be useful to avoid misunderstandings, specially if some words belong to the two idioms. The choice of a metaphor may be guided by learning, structuring or evoking power of each possible candidate following the main need.

It is also clear that humans are able to reason by manipulating symbolic representations that they translate afterwards into actions or compare with external events [25]. The direct manipulation tries to link mental manipulation and on screen symbolic manipulation. Its main characteristics are a continuous and visible representation of objects and actions sometime reversible, the help of physical actions and devices like buttons instead of complex command languages. The immediate consequence is an easier learning and a quicker remembering and usage of commands with more use of manual memory and less use of keyboard, syntax or grammar but the counterpart includes disadvantages like less complex requests [10], [1]. Not adjusted to all applications, it is often a goal instead of a criterion of judgement [6] but in virtual reality (V.R.), a *natural* interaction is one of our primary wishes.

The virtual reality is at the crossroads of 3D graphics, visualization, interaction, building of interfaces and metaphors, multimodality and direct manipulation. In the well known desktop metaphor or in other ones, we find some of these qualities but a major difference is that the user is far from the interface. We mean here that he had a sight on the « real » world and when he is implied in his task it is mainly mentally.

The V.R. deals with immersion, interaction and navigation. For immersive or not applications, the goal is to isolate the user in the computer generated world made realist, coherent, well suited and welcoming. If these requirements have been filled then the « presence » quality will be achieved too and a gain in performance should be clear. A V.R. explorer does not have the same facilities that a desktop user to work, he have some advantages but he lacks of perspective on his work and of comfort and he can not switch between tasks. So he is more prone to reject a bad interface.

Our personal experience of some category of metaphor of navigation has already proved that some metaphors are not pertinent for all applications. For example, simulating an object taken in the hand for exploration is good so long as you want to see its outer part and is better suited psychologically for fist sized objects such as a miniature model. Turning round a skull to examine is not practical too. « Flying » with V.R. software in a house with walls or inside a closed combustion chamber is not always convenient for different reasons be they logical or physical. The problems will surely be worst

and more complex if the software needs to manipulate some of its graphical objects, alternatives of navigation or action are sometimes the least worst solution. Each kind of activity like game, scientific visualization, architecture has its own particularities about navigation, immersion, exploration and interaction. A global and common solution has never existed, we must limit our goal to find a general way to solve the terms of a problem.

These reasons motivate a will to rationalise the choice of a metaphor, a process often empirical and not well described. With a good metaphor the learning is simpler, the use quicker and more implicit because the cognitive system relies more on the perceptive and motor system, the imagination is then unleashed. Using 3D new presentations like MagicCap for a desktop, a modified FSN to control the Jurassic Park Centre [24] or the navigations tool « Information Visualizer » from XEROX may be some kind of solution for problems of visualization. The selection, action, representation, navigation and dialogue with the metaworld are as many different needs sometimes conflicting to resolve in order to obtain a satisfactory final interface.

We present here a methodology helping in the conception and integration and test of metaphors more aimed for virtual reality. We define a list of points of interest and for each one general criteria of judgement.

CASE STUDY

In the goal of illustrating our subject, we will examine the problem of representation of the file system of a workstation. A computer user, advanced or not must be able to explore and to watch over a file hierarchy, to evaluate disk usage, to find out new files, to visualize or to execute them.

STAGES IN THE CONCEPTION OF A METAPHOR

Functional definition

Amongst innovations, we must identify which ones should be conveyed to the user, specially spirit of the program and *modus operandi*. The main questions expressed by the operator are well known [22]. The designer should be aware of their nature, then he will try to list some pertinent of them by advance and to propose some corresponding possible answers, some interesting first approaches will always be present.

Type of question

1) Goal oriented
2) Descriptive
3) Procedural
4) Interpretative
5) Navigational**Canonical form of the question**

1) What kind of things can I do with this program ?
2) What is it ? What does it do ?
3) How does it do this ?
4) Why has this happened ? What does it mean ?

5) Where am I ?These questions imply which kind of clues we have to give to the user. There are different models too, near if possible that must be developed such :

a) the targets system (discovered by the user) ;

b) the conceptual model of the target system (taught to the user) ;
c) the systems image (the feeling of the user) ;
d) the mental model that the user has of the target system ;
e) the conceiver's model of the users model.

Here, some important points are to show the hierarchical nature of our file system, the notion of name, location, size, owner, age and the possibility to consult, execute or search some file. The user must be helped and encouraged to browse the file system and to locate himself in it.

Participatory design

This point is a key element because a lot of misunderstandings or potential difficulties encountered by a user may be noticed earlier if some time is spent discussing with users about their specific knowledge and understanding, or testing in each stage the validity of possible solutions. The designer is the lone responsible for users mistakes, so using prototypes and benches, observing users testing a software or explaining themselves its use is an economic and proven but not sure way to limit ulterior disappointments. Most of the time, jobs are associated with co-operative tasks, precise times or places ; workers are aware of it and can give some useful clues.

The notion of hierarchy and the orientation may be a possible confusing point to a user not computer very literate, as the rights of access to some files.

First considerations

We must notice that the choice of a metaphor is partly implicit. Stating the problem, we use stylistic devices which expose relations between handled entities, informations flows, possibilities of action on them and their reactions, their organization... The problems encountered by the technical people are less pertinent here than the users ones. It is the ergonomes duty to highlight these points and to collect the good vocabulary. The choice of words is here very important, for example links and pipes have a different meaning about directionality [8]. However, the context is very different according to the type of application (game, visualization, simulation...). Users have often their own metaphors they teach themselves during the discovery of a job, they explain time, space or relations related problems. Each kind of problem has its primary requirements like interaction for games, navigation for visualization, realism for simulation. The choice of the metaphor of representation in the next stage will limit the other ones.

DIFFERENT METAPHORS

Metaphors of representation

First ideas. Some suggestions exist about the conception of mental models, we have natural ways to generate metaphors [25] : strong analogy, substitution, mapping, coherence, vocabulary, dimensionality of the problem, grammar, usage. Already aforesaid, our natural inclination is to align the characteristics or relations of objects in the virtual and real world. The spatial dimensions should reflect the semantic dimension of the task that users are confronted to. It is not possible to say that informations and their relations are convertible into a well known geometric, organizational or architectural ordering It could be good to notice similarities but as techniques as data are lacking. Establishing a diagram entities-relations as for a database and an object model may give some clues before looking in the users and our personal experience for possible candidates to a metaphor. We have in our case study an example of strict hierarchy with of lot of

file. So shapes, colours, symbols, sounds may be good candidates to classify things.

Here, the file structure is strongly and hierarchically organised. We find this property for example in our jobs, our towns or in a library. A working people may exercise some tasks in the same time, may be it is an opportunity to represent the link notion but an administration is rarely modified a lot. A city may illustrate a kind of hierarchy too, style and size of buildings giving clues about age and size. It can offer a global sight to this ordering, an unique opportunity. This metaphor has been retained in the « File System Navigator » [24] for this reason. At last, a library shows a good notion of document with different criteria a research but here it would be a richer library with text, video and other documents. Age or size may find a good fit with different colours of paper and with different thickness of books, the library seems the more convenient solutions.

Metaphors of selection and action

Selection. Voice, mouse, magical wand or any peripheral has its own advantages. The multimodal input is rich, redundant, attractive but far-off. Objects must show their selectionability by sight, sound or touch. The trend to give collectors and methods to objects will allow this dialogue [23]. Representing things in 3D space and without force feedback is a breach to security and confirmation. The comparison between open and closed loops, i.e. with tactile or force feedback, shows a 50% gain in time performance [11]. It must be remembered that man is polyvalent enough to replace a sense by another, like touch by sound.

Grammar of action. Two syntaxes are opposed, giving an object then the action used with or the opposite. The direct manipulation chooses the order object-action but natural activities make us take a tool then use it on the object. Choosing an object has the advantage to limit the choice of tools and inversely. Current actions are often two-handed but few people have studied an equivalent in computer dialogue [2]. The direct manipulation is inadequate to order a general action, an example being given [6].

Indication of possible actions. Appearance of the different objects of a scene and details of rendering are the first means to distinguish potential actors and decoration. Visual organization is another, a third being an event emitted by the object like colour or sound when it is in the focus of an action., the graphical pointer can change too. Ergonomics state that the shape indicates the function and the means of action [18] but by fortunately most of models do not require interactions [5].

For us, some documents (files) can not be seen by a user (access rights). The visual organization can show it, some books being too high in the shelves to be taken, behind a glass or forbidden to reading because they have a coloured pastille. Actions have the limits aforesaid i.e. searching with index or with the help of writings on the edge of books (author, rights, size...), consulting text or movies while placing the document on a lectern. Here the direct manipulation seems useful and is a natural way to do different actions.

Variety and representation of actions. Actions are shown by tools panels or multishapped cursors [2], [17]. Handled objects may be helpful, giving some help or clues about potential actions as a graphical handle signifying possible displacement. In an ideal case, each modification of the objects internal state should be reflected outside, be distinguishable and be reversible as far as possible [10]. The tools scale of sensibility must be fitted to the needed scale of action.

The hand is our usual tool to consult documents. We do not have a real need for a peripheral with six degrees of freedom because the books

in a shelf are always in the same vertical plane, a mouse may be sufficient. When the hand is near to a book, the index may be pointed to help the reading of the edge. To take a book, a button maintained pressed as long as needed is a good alternative to the rendering of the weight of the book. Some pertinent sounds (opening, taking...) will add some realism.

Metaphors of creation

Tools palette, modeller. Often, blank objects are presented to add them new components or to modify them with a tools palette. This case is quite similar to the precedent one about action. Sculpting shapes in a virtual world is a quite unexplored for the moment.

Duplication of existing objects. Unusual in real life, to have a signification, we must dispose of the metaphor of a duplicating machine, but to photocopy is not logical for each metaphor. Generally, the creation is an extension of manipulation and leaving the virtual world for the metaworld is often mandatory.

Here we have no need to create new books but if necessary we could add some blank books that could be filled writing, speaking, duplicating...

Metaphors of navigation

Completing the representation metaphor, this one is justified as soon as the quantity, nature or appearance of data is restricted by the display peripherals or if different view angles allow a better understanding. The right metaphor is the one leading to explore data.

Point of view. Ideally, the camera should be enslaved to the head, the body being its own vehicle, but the head is not very mobile. Seeing through someone elses eyes is a frequent cause of sickness or nausea. The world in the hand technique is psychologically better adapted to fist sized worlds. The eye in the hand way is precise too but can become uncomfortable quickly. Global performance are very changeable and depends on the global context.

Relocalization and orientation. Visual clues like different angular speed, occulting elements, having a global survey or judging distances by rotating the camera are less costly and more important ways of interpreting 3D relative positions than real stereoscopy. With his innate sense of orientation and some contextual indications disposed in key points, the operator is able to build his mental map of the virtual world. Techniques used by architects in modern buildings can be helpful for the virtual world designer [19]. He has the responsibility to fix an arbitrary arrangement if none exists. If the spatial organization is complex or at least in the first journeys in a virtual world, a graphical map has to be displayed [5].

Choice and reach of a destination. A way may be accomplished by teleportation, the destination being specified or a portal, hidden or not being stepped over [23]. Instead, the direction and target of smooth movement can be given, which is a practical solution. This may be specified by sights or hands direction, by a device or along a privileged direction [14]. The movement tied to the sights direction is probably the better for learners. The last way of giving a direction is the use of a physical activity like walk or cycling where an implicit direction is given [7]. Here too, the choice of a technique is strong tied to the nature of the exploration and the environment.

Speed control. Instantaneous movement is the privilege of teleportation, but on the other hand each time a mental map has to be rebuilt by reorientation. Adopting a speed varying logarithmically with the distance which remains is a very comfortable method, well suited for short or long movements [16]. The speed may be bound to a physical or symbolic action, following the rhythm of some device

activity. In the WalkThrough simulator, a bike allows to visit a building, the wheels being braked when going upstairs. This is a good example where a closed loop exists during the movement

Constraints on the movement. Living in a 3D world, the man builds mostly mental maps with two dimensions. In a building, we are changing floors rather than of altitude. A flight interface offers problems of co-ordination amplified by narrow lines of view and lack of gravity [28]. So, an artificial sky-line or far objects are needed to help our peripheral vision. In the same time, nearer and displayed objects bound to us like hands or feet limit the visual stress and lessen potential bad orientations effects [4]. Out of reach of the *vulgum pecus*, the control of 6 degrees of liberty in space has to be limited to 5 or less. For coherency, visual obstacles must make the movement physically more difficult or longer.

Medium. With the help of peripheral devices, inputs may be directions, gestures or rhythms [21]. A glove used, the hand gives direct, symbolic or constrained orders, discreet or continuous [27]. However common to pilot vehicles, using the feet has not really being studied, just in one case as a potential pointing device even if it frees hands or give a multimodal input [20]. Some actions helping fulfilment of the same task do not generate confusions, the performance is generally better and the presence strengthened. Physically implying, the walk may be simulated, limited by obstacles, staying in the same place or unlimited with the aid of a travelling band [7]. Replicas of different pilots cockpits give consistent interfaces but generally not well suited for actions metaphors. Virtual cockpits are useful to define future prototypes but often appear of no use for real driving, at least by lack of a force-feedback [26].

> *In our library, the natural candidate for the movement is the walk, at last for logical and practical reason. To help the explorer, the opening of a folder may be translated in a walk to new shelves. Instead of a walk, a teleportation with a soft transition of images and a sound of steps may signify this displacement, a map helping to orientate oneself or to go to a precedent location. If we need a global sight, a different geographic organization is requested. The research of files has a substitute in filling a research card.*

Metaworld

Disengagement. The V.R. faces the specific question of disengagement when being immersed in the virtual world. It can be to access to a keyboard or in a augmented reality system to see just the real environment. Prepared for, the device drivers have the responsibility to judge this intention and to inform the main program. Exterior events like getting too close to a wall must be signified. Here, the real world enters in the virtual worlds.

Inputs, outputs, exchanges. The definition of the metaworld makes this word represents all which is not the application itself. In an editor, we toggle between command or edit modes. Here it is the same, we can need to give commands not possible otherwise, to modify the appearance of our interface or to exchange documents with the outside. The ultimate goal should be to not have to let the inner side of a metaphor except to use an other software. The clipboard in a Windows-like environment allows to exchange data but most of the time the limits of the metaphors are blurred and reconfiguration or external activities are lowly consistent. In a desktop metaphor, modifying the colours or some parameters may be resolved with a paper catalogue of desk accessories which allow the reconfiguration without « leaving » the context of the desktop environment. Metaphors

are not auto inclusive without difficulties but some tricks can mitigate the consequences of this limitation.

In our library metaphor, the graphic representation can accept different rooms for different activities and a small wagon for exchanging data (name or content of files) with external activities. Switching jobs implies to pass doors or leaving our library to turn off the light switch.

GLOBAL COHERENCE

Ergonomics. This stage verifies that the whole actions are well enumerated, easy to learn or to remind and easily distinguishable except in case of strong analogies.

Scale factors. Time lengths and delays, distances must be evident and possibly the same in each metaphor. Real or differed time, in place, distant and other scale actions are pertinent informations and should be reflected clearly.

Environment. If possible, a majority of the objects which are present in the real world that the metaphor represents must be here. The virtual world will offer a complete picture and the presence will be higher. Fundamentally different concepts, scales of space or time that are too large, objects too technologically distant are real obstacles to the global coherence. The V.R. needs specially a excellent coherence because the user is very « close » to his interface and can be rebuffed much more easily.

The initial goals are fulfilled with natural, instant and reduced interactions with familiar objects in the common environment of a library with all its usual features.

CRITERIA TO EVALUATE METAPHORS

For each proposition, we have to measure the functionalities along some criteria which are very application-type dependent. Some criteria proposed elsewhere were insufficient and not satisfactory for the needs of V.R. We had to extend or modify them [8], [12].

Coherence. It measures the clearness of actions and their consequences. We need to guess an internal logic and strong relations between action on the metaphor and internal modifications of data.

Presence. When the operator finds everything he needs in the virtual world, if all his senses are fed and work together in a coherent universe then this ultimate goal is reached, the virtual world becoming the real world.

Malleability. It is possible to configure differently or to enrich the interface, being not ordered to follow a unique way of doing but instead being able to define ones own methods.

Participation and interactivity. As a real life action which uses all our senses and our concentration, a metaphor must imply full participation. A game-like aspect can serve the interactivity. Moreso, the exchange is directed to human being.

Personality. The spirit, aspect or other qualities of a program may be signed with a stamped signature which make it different from others. It is sometimes a criterion of choice and remembering.

Adequation. Nothing is worse than to have to decrypt the usage of a tool. A metaphor must tell us what we are doing here and how to do it. Helping to fulfil a work, the clearness is gratifying.

Learning. Any metaphor must be based on the knowledge of an individual being accustomed to the task but with a general knowledge and no particularisms be they cultural or technical. Progressive and pleasant, the learning must build mental models that are more and more abstract.

Representation. Even perfect, a program must adapt itself with the limits of the current technology. For example, a simple but reactive interaction is better than a slow but detailed one. A device driver taking to account the means of rendering should adapt the real possibilities of the software to those of the hardware.

Ergonomics. Mentioned more precisely here for V.R., we must limit the physical, visual and mental tiredness. The actual conditions of V.R. are more restricting than anywhere else.

Usage and possibilities of evolution. Recurring operations are to lighten or to banish. A good metaphor is fully exploited and adequate. More, it can adapt without too much conditions to changes or additional request to the terms of the problem.

> *This library style interface seems adequate to most of the initial terms and will be comfortable if the interface is clean and smooth. We can ask ourselves if this metaphor can accept additional requirements like file rightss modification, files move or deletion, use of filter of research or display... It seems that a lot of options are allowed here, the metaphor has still some potential.*

CONCLUSION

A global survey of the elaboration of our library metaphor has highlighted some requisite points not stated in the first terms of the problem. The need for a common metaphor, for criteria and short cuts of research, for possibilities of evolution has been shown. The result shows a good fulfilment for navigation, selection and interaction. The will to avoid the miss of orientation or to allow exchanges with external jobs are clear now. The user of our library has no use of a metaworld except to do other jobs. There is no distinction between two modes like « editing » or « commands », it is an excellent thing because this metaphor is self sufficient.

The *modus operandi* discussed above aims to be general but complete, we believe we have made a step towards our goal. This methodology was proposed to help to conceive and to integrate metaphors. It is better suited to the particular nature of virtual reality than other methodologies because different needs like navigation, creation or others are evoked. Along these stages of the elements of a problem, the conceiver gains a better understanding of important points that he want to communicate. Nevertheless, the adoption of a program and its interface is often a case of personal taste. To palliate this, a participative conception all along this work is essential to prevent misunderstandings, easily confusing the user.

FUTURE WORK

In order to characterise further action and the judgement *a priori* or *a posteriori* of the adequation of metaphors with statements, the following points appear important :

- measuring and defining clearly the base characteristics of the presence, relations between feel of presence and qualitative or quantitative performance of a user ;
- studying relationships between perceptual space, cognitive space and degrees of freedom of the peripherals, extending of metaphors to more than three dimension problems ;
- improving in feedback, substituting to force feedback, using sound and voice.

REFERENCES

[1] J.A.Ballas, C.L.Heitmeyer, M.A.Pérez, "Evaluating two aspects of direct manipulation in advanced cockpits", CHI'92
[2] E.A.Bier, M.C.Stone, K.Pier, W.Buxton, T.D.Derose, "Toolglass and Magic Lenses : The See-Through Interface", Computer Graphics Proceedings, Annual Conference Series, 1993
[3] M.Bordegoni, M.Hemmje, "A dynamic gesture language and graphical feedback for interaction in a 3D user interface", Eurographics 1993
[4] M.Bricken, "Virtual worlds : No interface to design", IMAGINA'92, I42-I57
[5] F.P.Brooks, "Grasping reality through illusion : Interactive graphics serving science", CHI'88, pp 1-11
[6] B.Buxton, "HCI and the inadequacies of direct manipulation systems", SIG'CHI bulletin, vol. 25 N°1, 1/1993
[7] C.Cruz-Neira, D.J.Sandin, T.A.Defanti, R.V.Kenyon, J.C.Hart, "The Cave : Audio-visual experience automatic virtual environment", Communications of the ACM (6/92)
[8] T.D.Erikson, "Working with interface metaphors", in "The art of human computer interface design", Brenda LAUREL, Addison Wesley, 1992, pp 65-73
[9] O.Dupuy, "De l'usage des pieds", Internal paper, C.I.G., Institut Gaspard Monge, Université de Marne la Vallée, 1994
[10] R.E.Eberst, K.P.Bittianda, "Preferred mental models for direct manipulation and command-based interfaces", p 769-785, International journal of man-machines studies (**38**), 1993
[11] C.Esposito, "Virtual reality : Perspectives, applications, and architecture" in "User interface software" de Bass & Dewan, 1993 John Wiley & Sons Ltd
[12] B.Jacobson, "The ultimate user interface", BYTE 4/92, pp 175-182
[13] K.Fairchild, G.Meredith, A.Wexelblat, "The tourist artificial reality", CHI'89, pp 299-303
[14] M.Fleischmann, "The Cyber City Project", IMAGINA'92, pp IV6-IV11
[15] T.A.Funkhouser, C.H.Séquin, "Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments", SIGGRAPH'93 Proceedings
[16] J.D.MacKinlay, G.G.Robertson, S.K.Card, "Rapid controlled movement through virtual 3D workspaces", CHI'91, pp 455-456
[17] M.J.Muller, "Multifunctional cursor for direct manipulation user interfaces", CHI'88
[18] D.Norman, "The design of everyday things", Doubleday currency (NY, Londres) (a Banthams division)
[19] R.Passini, "Way finding in architecture", Van Nostrand Rheinhold, 1984
[20] G.Pearson, M.Weiser, "Exploratory evaluation of a planar foot-operated cursor-positioning device", CHI'88
[21] D.Rubine, "Specifying gestures by example", Computer Graphics, vol. 25, n° 4, 7/1991
[22] A.Sellen, A.Nicol, "Building user-centered on-line help", in "The art of human computer interface design", Brenda LAUREL, Addison Wesley, 1992
[23] Sense8 Corporation, Sausalito, CA, "WorldToolKit Reference Manual", 1993
[24] Silicon Graphics, binairies of FSN with ftp on ftp.sgi.com
[25] N.Staggers, A.F.Norcio, "Mental models : concepts for human-computer interaction research" , p 587-605, International journal of man-machines studies (38), 1993

[26] R.J.Stone "The UK VERDEX project, 2 years on : VR, telepresence and the human factor", IMAGINA 92, pp IV35-IV52
[27] David Joel Sturmann, "Whole hand input", Ph.D. Thesis, Media Arts & Sciences Section, M.I.T., 1992
[28] C.Ware, S.Osborne, "Exploration and virtual camera control in virtual three dimensional environments", ACM Symposium on interactive computer graphics 90, pp 175-183

Improving the Legibility of Virtual Environments

Rob Ingram and Steve Benford

Department of Computer Science
The University of Nottingham
Nottingham, NG7 2RD, UK

Tel.: +44 602 514203
Email: s.benford, r.ingram @cs.nott.ac.uk
Fax: +44 602 514254
http://www.crg.cs.nott.ac.uk/

1. Introduction

Years of research into hyper-media systems have shown that finding one's way through large electronic information systems can be a difficult task. Our experiences with virtual reality suggest that users will also suffer from the commonly experienced "lost in hyper-space" problem when trying to navigate virtual environments.

The goal of our paper is to propose and demonstrate a technique which might help overcome this problem. Our approach is based upon the concept of *legibility*, adapted from the discipline of city planning. The legibility of an urban environment refers to the ease with which its inhabitants can develop a cognitive map over a period of time and so orientate themselves within it and navigate through it [Lynch60]. Research into this topic since the 1960s has argued that, by carefully designing key features of urban environments planners can significantly influence their legibility.

Our paper proposes that these legibility features might be adapted and applied to the design of virtual environments of all kinds and that, when combined with other navigational aids such as the trails, tours and signposts of the hyper-media world, might greatly enhance people's ability to navigate them. In particular, the primary role of legibility would be to help users to navigate more easily as a result of experiencing a world for some time (hence the idea of building a cognitive map). Thus, we would see our technique being of most benefit when applied to long term, persistent and slowly evolving virtual environments. Furthermore, we are particularly interested in the automatic application of legibility techniques to information visualisations as opposed to their relatively straight forward application to simulations of the real-word. Thus, a typical future application of our work might be in enhancing visualisations of large information systems such the World Wide Web.

Section 2 summarises the concept of legibility as used in the domain of city planning and introduces the key features that have adapted in our work. Section 3 then describes a set of algorithms for the automatic creation or enhancement of these features within virtual data spaces. Next, section 4 presents two example applications based on two different kinds of virtual data space. Finally, section 5 presents some initial reflections on this work and discusses the next steps in its evolution.

2. What is legibility?

Legibility, in the context of navigation and wayfinding, is a term which has been used for many years in the discipline of City Planning. Work on legibility in this area has been

concerned with the way in which people are able to 'read' an environment and hence perform wayfinding tasks. In his book "The Image of the City" [Lynch60] Kevin Lynch defines the legibility of a city as: "...the ease with which its parts may be recognised and can be organised into a coherent pattern..." Here, Lynch is referring to the formation of a *cognitive map* within the persons mind [Passini92], a structure which is an internal representation of an environment which its inhabitants use as a reference when navigating to a destination. The Image of the City describes experiments carried out in a number of major US cities which show how the cognitive map is built up over time through experience of the city. The experiments involved obtaining information from long term inhabitants of the cities in the form of, for example, interviews, written descriptions of journeys through the city and drawn maps. By examining this data Lynch identified five major elements of urban landscapes which are identified by the inhabitants and used as the building blocks of the cognitive maps. These features are:

- **Landmarks.** Static and recognisable objects which can be used to give a sense of location and bearing
- **Districts.** Sections of the environment which have a distinct character which provides coherence, allowing the whole to be viewed as a single entity
- **Paths.** Major avenues of travel through the environment such as major roads or footpaths
- **Nodes.** Important points of interest along paths, e.g. road junctions or town squares
- **Edges.** Structures or features providing borders to districts or linear obstacles

3. Legibility techniques for virtual environments

The aim of ongoing research at Nottingham University is to apply the work described above not to real environments but to the artificial spaces of virtual reality systems. More specifically we are developing techniques to automatically construct the five legibility features in the abstract spaces produced by data visualisation systems such as database or document store visualisers. One of our main aims is to accomplish this with without requiring the users of the system to perform the placement of the features manually. Essentially the system should, wherever possible, identify and place the features using information available from the database and visualisation systems alone.

To do this we have constructed a prototype system called LEADS (LEgibility for Abstract Data Spaces) which is designed to provide a layer on top of existing visualisation systems and which performs the addition of legibility information to the space. LEADS acts on the position of data items provided by this underlying system, as well as accessing the raw data where necessary, to add and emphasise the objects which are used to improve the legibility of the environment.

LEADS is designed to be applied to spaces which satisfy three main criteria. They should: be persistent over relatively long periods of time; be relatively stable so that they evolve over their lifetime and are rarely disturbed by major upheavals in the database; be accessed repeatedly by a number of independent users. An example of a large space to which application of LEADS techniques might be appropriate is the WWW space, which is constantly evolving but which rarely undergoes global scale restructuring.

To place the legibility features LEADS uses districts as a starting point as this allows for a number of relatively simple techniques to be used to form the other features. Districts in a data space will be clusters of items which have some sort of internal similarity. LEADS

applies a clustering algorithm to the data to identify these groups automatically. The algorithm chosen for the initial implementation is Zhan's Minimum Spanning Tree Algorithm [Zhan71], which works by forming a minimal spanning tree of the distances between the data items and then walking the tree to remove links which are significantly longer than others nearby. The sub-trees resulting from the use of the algorithms each form a district in the space.

Landmarks need to be placed in a position where they will be useful for navigation. We have used a premise that one such place will be where districts are dense. The first step in positioning a landmark is to find groups of three clusters which are mutually adjacent. A landmark will be placed in a position which is fairly central to these districts. This is found by finding the centroids of each district and placing the landmark at the centre of the triangle they form.

Edges in LEADS are structures which help to define the borders of districts. They are placed in the space between those districts which are significantly large. To accomplish this they are placed between the nearest neighbours in the cluster and aligned along the line that joins them. In most cases, especially where the clusters are essentially spherical, this results in an edge placement which effectively separates the space but does not cut into the individual districts.

Nodes and paths are co-dependent in the LEADS system. Eventually the aim is to have these features evolving out of the use of the data space. The initial prototype currently identifies nearest neighbours between districts as nodes and places a path between them. Within districts all those items identified as nodes are joined by a spanning tree.

4. Two example applications

We have so far applied LEADS to the visualisation produced by two existing information visualisation tools. The first of these, Q-PIT, is a system which works on databases where the items have a number of well defined fields [Benford94]. Three of these fields may be chosen so that the values they contain are mapped onto the three major axes of the space to give the position of the data items. The remaining fields may be used to define aspects of the representation of the items such as their shape, colour or speed of rotation.

The second system, Grapher, is a 3D graph drawing tool. This takes a representation of an arbitrary network and produces a 3D visualisation by representing the nodes as balls and the links as springs. Initially the items will be placed randomly and the system will then go through a cycle of repositioning the nodes based on the tension in the springs until a relatively stable formation is found.

LEADS attempts to enhance the legibility of both systems through the above techniques. Figure 1 shows before and after shots from Q-PIT. Figure 2 shoes before and after shots from Grapher[1].

5. Reflections

Our initial experiences of developing and testing LEADS have been largely positive and we suspect that this approach does have the potential to significantly improve the legibility of virtual environments. However, we have encountered a number of difficulties which suggest some immediate improvements to the system:

First, legibility features such as paths seem to sit uncomfortably with 6 degrees of

1. Because of the extensive use of colour in LEADS these plates can not fully demonstrate the effectiveness of the enhancement. We hope however that the do give some idea of the effect of the addition of legibility features.

freedom navigation. Paths in a city are actually travelled along and the environment is therefore experienced from the perspective of the path. We suspect that users of our legible virtual environments should also directly experience paths as part of navigation. Thus, we need to develop simple interface techniques to support navigation via paths (e.g. "take me along the nearest path in this direction").

Second, automatically determining an appropriate scale and appearance for legibility features has not always been easy. In particular, features such as landmarks and edges must be visible without being intrusive. Creating useful edges has proved to be a particularly difficult task as an edge should ideally follow the contours of a district. In a 3-D space, determining a sensible size for edges has been difficult. At one extreme an edge might be a hull completely surrounding a district. At the other it might be a thin flat surface dividing two districts. The former is likely to be visually intrusive; the latter is likely to provide an insufficient sense of separation between districts.

Third, other features and tools are clearly needed to help people navigate. For example, the use of textual information in the form of signposts is an important part of navigating conventional urban environments. We can imagine adding signposts to virtual environments and placing them at nodes pointing along paths. Furthermore, we can imagine that signposts might refer to districts and landmarks that lie along a path. However, this gives rise to the problem of how to name districts and landmarks. More specifically, given that districts and landmarks are automatically created by LEADS, we are left with the problem of automatic name generation or alternatively the use of non-textual symbolic identifiers on signposts.

Finally, we need to be careful that by adding additional objects to an information visualisation, we do not increase the rendering overhead thereby degrading system performance. However, we suspect some extensions to LEADs might enable it to actually improve system performance. Specifically, LEADS might enable a general distancing effect for data spaces by allowing the individual objects in a district to be replaced by an overall representation of the district when viewed from a distance.

6. Summary

We have described a number of general techniques for improving the legibility of virtual environments so that their users might more easily construct cognitive maps to help them navigate. Our work has adapted techniques from the discipline of city planning where decades of experience have identified key features of urban landscapes which are critical to their legibility. The primary goal of our work has been to develop a set of algorithms for *automatically* creating or enhancing these features within information visualisations. These include:

- the use of clustering algorithms to create districts;
- the creation of edge objects separating districts;
- the placement of landmark objects at a central point between districts;
- emphasising nearest neighbour node objects within districts and creating paths between them.

We have implemented these techniques in a system called LEADS which is intended to provide an additional legibility layer sitting on top of current information visualisations. So far, we have applied LEADS to two existing and contrasting information visualisation tools. Our paper included some before and after screen-shots to show the overall effect of LEADS on these systems.

Our early experiences have been positive and suggest that this approach is promising.

However, we have encountered several difficulties including the need to experience paths when navigating; the difficulty of getting an appropriate scale for features such as edges and landmarks; and problems with automatically naming features so that they may be referred to by signposts.

Longer term work will include combining LEADS with larger scale, more widely used information sources (e.g. visualising the World Wide Web) and carrying out a more formal evaluation of our work. For the latter, we intend to revisit and adapt the initial experiments carried out by Lynch within real cities, but this time within virtual environments.

7. References

[Benford94] Steve Benford, John Bowers, Lennart Fahlén, Chris Greenhalgh, John Mariani and Tom Rodden, *Networked Virtual Reality and Co-operative Work*, Presence, MIT Press, (in press).

[Lynch60] Kevin Lynch, *The Image of the City*, M.I.T. Press, 1960

[Passini92] Remedi Passini, *Wayfinding in Architecture*, Van Nostrand Reinhold, 1992

[Zahn71] Zahn, C.T., *Graph-theoretical Methods for Detecting and Describing Gestalt Clusters*, IEEE Transactions on Computers, C 20, 68-86

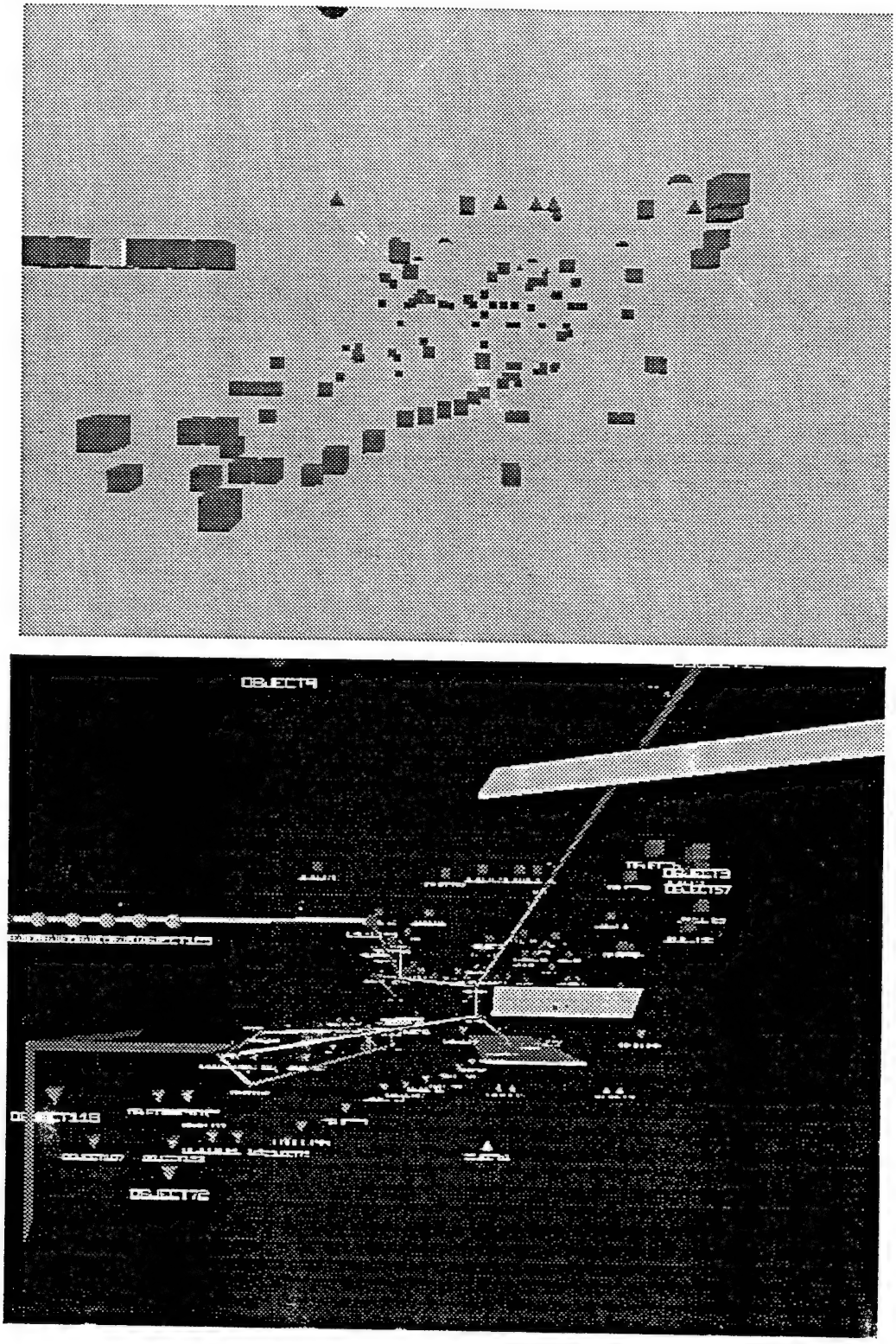


Figure 1: Q-PIT before (top) and after (bottom) legibility enhancement using LEADS

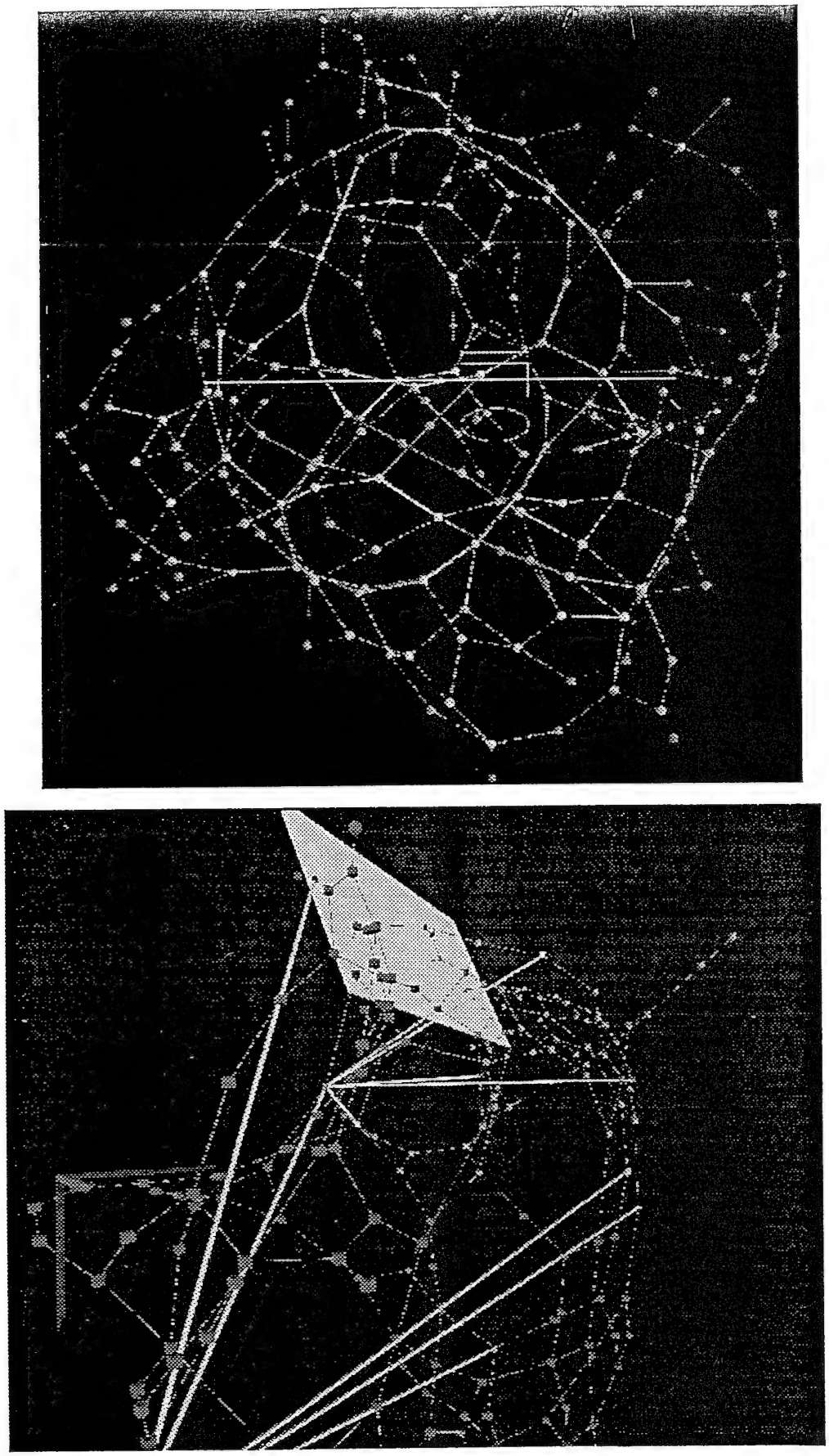


Figure 2: Grapher before (top) and after (bottom) legibility enhancement using LEADS

Fast algorithms for drawing Nonuniform B-spline surfaces: a practical application in Virtual Environment

M. Bergamasco

PERCRO
Simultaneous Presence, Telepresence and Virtual Presence
Scuola Superiore S. Anna
Via Carducci, 40, 56127 Pisa, Italy
e-mail: bergamasco@sssup1.sssup.it

D. Bucciarelli, P. Degl'Innocenti

Humanware s.r.l
Via XXIV Maggio 62, 56127 Pisa, Italy
Tel/Fax +39-50-554108

Abstract

The paper we intend to present deals with the problem of drawing parametric surfaces in real-time. A new fast sequential and parallel algorithm for approximate parametric surfaces with polygons is described. Its application on a single and multi-processor system is presented by emphasizing the constraints imposed by real-time applications for Virtual Reality. In particular a practical utilization of parametric surfaces for the modeling of a virtual hand-arm complex is described.

1 Introduction

The requirements of real-time performances can be considered one of the irremissible aspects to be taken into account when the control of a direct interaction with Virtual Environments must be obtained. This fact assumes even greater importance for those applications tied with manipulative procedures, in which the human operator is asked to touch, explore and grasp virtual objects according to a realistic behaviour [7] [8].

The constraint of achieving a high degree of realism in the performance of such operations involves all afferent sensory modal pathways of the interface systems. In particular, as far as the visual component is concerned, the intense utilization of high level rendering techniques for the graphical representation of the objects and human body parts belonging to the Virtual Environment must be considered. In this paper we address the problem of achieving an adequate real-time graphical representation of human body parts by exploiting a new algorithm based on Nonuniform B-spline surface descriptions.

As a theoretical introduction to the problem is given in the following by means of a brief description of the definition of nonuniform B-splines. For a complete introduction to the theory of parametric surfaces representation refer to [1][2][3][5][4].

If np is the number of control points and k is the order of nonuniform B-splines, the nonperiodic knot vector, for a curve or surface with the end points coincident with the end control points, is:

$$T = (t_0, \cdots, t_{np+k}) = (\underbrace{0, \cdots, 0}_{k}, t_k, \cdots, t_{np}, \underbrace{1, \cdots, 1}_{k})$$

Remember that we have to define only one knot vector with a curve, while, in the case of a surface, we have to define two knot vectors, one for each component. The blending functions are:

$$N_{i,1}(u) = \begin{cases} 1 & \text{if } t_i \le u \le t_{i+1} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$N_{i,k}(u) = \frac{(u - t_i)N_{i,k-1}(u)}{t_{i+k-1} - t_i} + \frac{(t_{i+k} - u)N_{i+1,k-1}(u)}{t_{i+k} - t_{i+1}} \quad (2)$$

The general formula for evaluating a parametric surface, of n per m control points, in a point (u, v), is:

$$\boxed{Q(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} p_{i,j} N_{i,k_u}(u) N_{j,k_v}(v)} \quad (3)$$

where k_u and k_v are the order in u and v of the surfaces.

For evaluating the efficiency of each alghoritms, we define:

$$Cost_Q(n, m, k) = nmCost_N(k) \quad (4)$$

where

$$Cost_N(k) = \begin{cases} 1 & \text{if } k = 0 \\ 4 + 2Cost_N(k-1) & \text{otherwise} \end{cases}$$

as the number of multiplications for evaluating the parametric surface in (u_{index}, v_{index}) with the function Q.

2 The algorithm

If we fix a set S of ns sample points:

$$S = \{(u_0, v_0)_0, \cdots, (u_{ns-1}, v_{ns-1})_{ns-1}\} \quad (5)$$

we can define a new function:

$$NN_{i,j}(index) = N_{i,k_u}(u_{index}) N_{j,k_v}(v_{index}) \quad (6)$$

We can now define the matrix $NT_{n \times m \times ns}$ (N Table) with $NT_{i,j,index} = NN_{i,j}(S_{index})$; for example the 2D matrix $NT_{i,j,1}$ is equal to:

$$\begin{Bmatrix} NN_{0,0}(S_1) & NN_{0,1}(S_1) & \cdots & NN_{0,m-1}(S_1) \\ NN_{1,0}(S_1) & NN_{1,1}(S_1) & \cdots & NN_{1,m-1}(S_1) \\ \cdots & \cdots & \cdots & \cdots \\ NN_{n-1,0}(S_1) & NN_{n-1,1}(S_1) & \cdots & NN_{n-1,m-1}(S_1) \end{Bmatrix}$$

The function (3) becomes:

$$Q(u, v) = Q_S(index)$$

where:

$$\boxed{Q_S(index) = \sum_{i=0}^{n} \sum_{j=0}^{m} p_{i,j} NT_{i,j,index}} \quad (7)$$

At this point we can easily precalculate NT and evaluate the parametric surface in (u_{index}, v_{index}) with only nm multiplications. As for (4) we can define

$$Cost_{Q_S}(n, m, k) = nm \quad (8)$$

The function $N_{i,j}$ (and the value of $NT_{i,j}(index)$ which is function of N) is equal to zero for many values of i and j. For (1) and (2) the function $N_{i,k}(u)$ (and for (6), the function $NN_{i,j}(index)$) is zero $\forall u \notin [t_i, t_{i+k}]$ and $\forall v \notin [t_j, t_{j+k}]$. This suggests another improvement for the algorithm. We can define the matrix $RIT_{ns \times 2}$ (Row Index Table) with $RIT_{index,0}$ equal to:

$$Min\{0 \le i < n \quad | \quad \forall \quad 0 \le j \le m, NT_{i,j,index} \neq 0\}$$

and $RIT_{index,1}$ equal to:

$$Max\{0 \le i < n \quad | \quad \forall \quad 0 \le j \le m, NT_{i,j,index} \neq 0\}$$

and the matrix $CIT_{ns \times n \times 2}$ (Column Index Table) with $CIT_{index,i,0}$:

$$Min\{0 \le j < m \quad | \quad NT_{i,j,index} \neq 0\}$$

and $CIT_{index,i,1}$:

$$Max\{0 \le j < m \quad | \quad NT_{i,j,index} \neq 0\}$$

CIT is defined only for $RIT_{index,0} < i < RIT_{index,1}$. The function (7) becomes:

$$\boxed{FQ_S(index) = \sum_{i=RIT_{index,0}}^{RIT_{index,1}} \sum_{j=CIT_{index,i,0}}^{CIT_{index,i,1}} p_{i,j} NTi, j, index} \quad (9)$$

The evaluation of the parametric surface in (u_{index}, v_{index}) now it costs:

$$(RIT_{index,1} - RIT_{index,0})(CIT_{index,i,1} CIT_{index,i,0}) \quad (10)$$

multiplications. The value of (10) is a function of the knot vectors (the function $N_{i,k}(u)$ is zero outside the range $[t_i, t_{i+k}]$) and it is not easy to estimate the value of $Cost_{FQ_S}$. For simplicity we can use this definition:

$$Cost_{FQ_S}(n, m, k) = \alpha nm \quad \text{where} \quad 0 < \alpha < 1 \quad (11)$$

with α as a function of knot vectors of the Nonuniform B-spline. Normally α is mainly a function of the order of the parametric surfaces. We can now evaluate the relative efficiency of our new algorithm with respect to classic algorithm:

$$E = \frac{Cost_Q(n, m, k)}{Cost_{FQ_S}(n, m, k)} = \frac{Cost_N(k)}{\alpha} \gg 1$$

In a practical case E is greater than 180-200.

3 The multi-processor evolution of the algorithm

The algorithm described in Section 2 is sequential but can be easily extended for direct use on a parallel machine. There is the theoretical possibility of implementing the algorithm on a massive SIMD parallel computer and on a MIMD parallel machine with local memory, but we

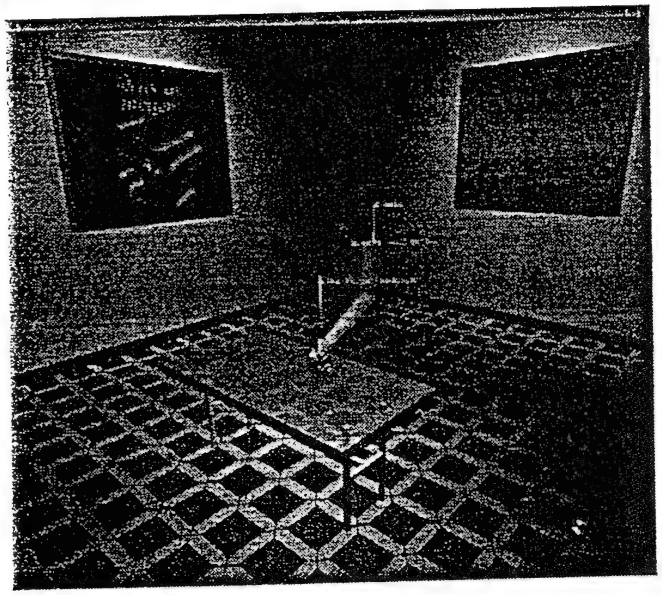


Figure 1: A B-Splines based virtual Hand/Arm used for force feedback experiments at Scuola Superiore S.Anna.

will describe only a theoretical and practical method for using parametric surfaces on a symmetric multi-processor system with shared memory and with a small number of processors, because this type of computer is the more diffuse and available kind of parallel machine. On a symmetric multi-processor system with shared memory and a small number of processors, the cost of communication and synchronization suggests to calculate more parametric surfaces in parallel than to calculate a single surface on multiple CPU. For writing a library with a C-like language in order to calculate nonuniform B-spline in parallel, we suppose to have the following function:

- a function *name of semaphore*=InitSem(*Initial value of the semaphore*);
- a function V(*name of semaphore*) for doing a signal on a semaphore;
- a function P(*name of semaphore*) for doing a wait on a semaphore;
- a function SProc(*name of a function*), a variant of the UNIX function fork, for creating a new process that is a clone of the process that called SProc and that shares the virtual address space of the parent process.

The shared dates of the library are:

```
#define MAX_BUF_SIZE 40

semaphore ProdSem,CalcedSem,MutexSem;

float ****NTable;
int ***RIT,****CIT;

ParamSurfacePtr PSurface[MAX_BUF_SIZE];
PolygonGridPtr SurfBuffer[MAX_BUF_SIZE];
NormalGridPtr SurfNormalBuffer[MAX_BUF_SIZE];

short SurfaceInBuffer,
  SurfaceCalced,NumSubdiv[2];
```

where *NTable* is a table of values defined by the (6). Please note that in a practical implementation, the set S, defined in (5), is a regular grid of values between $(0,0)$ and $(1,1)$:

$$NTable[i][j][k][t] = N_{i,k_u}(\frac{k}{NumSubd_u - 1})$$
$$N_{j,k_v}(\frac{t}{NumSubd_v - 1})$$

where $NumSubd_u$ and $NumSubd_v$ are the numbers of subdivision of the interval $[0 \cdots 1]$ in u and v. In the library, NumSubdiv[0] will be equal to $NumSubd_u$ and NumSubdiv[1] to $NumSubd_v$. The set S defined in (5) now becomes:

$$S_{k,t} = \left\{ \begin{array}{lll} (0,0) & \cdots & (0,1) \\ (\frac{1}{NumSubd_u - 1}, 0) & \cdots & (\frac{1}{NumSubd_u - 1}, 1) \\ \cdots\cdots & \cdots\cdots & \cdots\cdots \\ (1,0) & \cdots & (1,1) \end{array} \right\}$$

RIT and *CIT* will be initialized according to the definitions given in Section 2. The meaning of *ProdSem, CalcedSem, MutexSem, SurfaceInBuffer, SurfaceCalced* will be explained below. The initialization function, that is called only at the startup of the program, is:

```
void InitBSpline(short NumCtrlPnt[2],
  float *KnotVectorU,short NumKVU,
  float *KnotVectorV,short NumKVV,
  short NumSubd[2],short NumProc)
{
  int i;

  NumSubdiv[0]=NumSubd[0];
  NumSubdiv[1]=NumSubd[1];

  NTable=AllocAndInitNTable(NumSubd,
    KnotVectorU,NumKVU,
    KnotVectorV,NumKVV,NumCtrlPnt);

  RIT=AllocAndInitRIT(NumCtrlPnt,
    NumSubd,NTable);
```

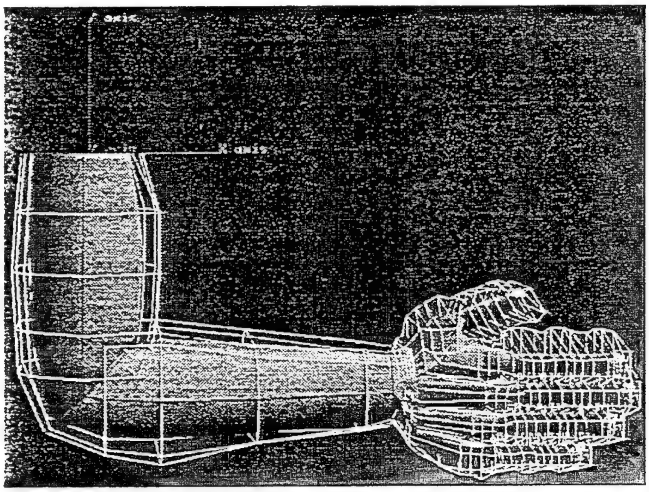


Figure 2: The virtual Hand/Arm and the grids of control points of the parametric surfaces.

```
  CIT=AllocAndInitCIT(NumCtrlPnt,
    NumSubd,NTable);

  ProdSem=InitSem(0);
  CalcedSem=InitSem(0);
  MutexSem=InitSem(1);

  SurfaceInBuffer=SurfaceCalced=0;

  for(i=0;i<NumProc;i++)
    SProc(SurfProc);
}
```

In this function we precalculate *NTable*, *CIT* and *RIT* as described in Section 2. The semaphore *ProdSem* is created and initialized to 0; on this semaphore the parent will signal to the children processes, generated by the function *SProc*, when a new nonuniform B-spline is present in the buffer *PSurface*. The semaphore *CalcedSem* is created and initialized to 0; on this semaphore the children processes will signal when they finished the calculus of the transformation between parametric surface and the grid of polygons. The semaphore *MutexSem* is used for mutual exclusion by the children processes. *SurfaceInBuffer* is the index of the first free cell in *PSurface* and it is equal to the number of surfaces in the buffer. *SurfaceCalced* is the index of the first parametric surface in the buffer *SurfaceInBuffer* to be calculated. The code of the library is:

```
void SurfProc(void)
{
  int x,y,cn;

  for(;;) {
    P(ProdSem);
    P(MutexSem);
    cn=SurfaceCalced;
    SurfaceCalced++;
    V(MutexSem);

    for(x=0;x<NumSubdiv[0];x++)
      for(y=0;y<NumSubdiv[1];y++)
        FastEvalP(PSurface[cn],
          x,y,SurfBuffer[cn][x][y]);

    CalcSurfaceNormal(SurfBuffer[cn],
      SurfNormalBuffer[cn]);

    P(CalcedSem);
  }
}

void CalcSurfacePoint(PSurfacePtr cp,
  PolygonGridPt pg,NormalGridPtr ng)
{
  PSurface[SurfaceInBuffer]=cp;
  SurfBuffer[SurfaceInBuffer]=pg;
  SurfNormalBuffer[SurfaceInBuffer]=ng;
  SurfaceInBuffer++;

  V(ProdSem);
}

void WaitCalcSurface(void)
{
  int i;

  for(i=0;i<SurfaceInBuffer;i++)
    P(CalcedSem);

  SurfaceCalced=SurfaceInBuffer=0;
}
```

The function *FastEvalP()* is the implementation of the formula (11). In *SurfProc()* we call the function *CalcSurfaceNormal()* for calculating the normals to the surface *SurfBuffer[cn]*. A typical use of the functions in the library is the following:

```
main()
{
    ...
  PSurfacePtr cp[NS];
    ...
  InitBSpline(...);
    ...
  for(;;) {
    CalcSurfacePoint(cp[0],...);
      ...
    CalcSurfacePoint(cp[NS-1],...);
      ...
```

```
    /* Do everything you want. */
      ...
    WaitCalcSurface();
    DrawSurfaces(cp,NS);
  }
}
```

4 Modeling a virtual hand and arm by parametric surfaces

A set of 40 nonuniform B-splines was used to design a virtual arm and hand. We successfully use our mono and multi-processor alghoritm for converting parametric surfaces in polygons meshes for a practical application. We conduct experiments using an *Arm Exoskeleton System* (see [6]) and a *Hand Exoskeleton System* developed at Scuola Superiore S. Anna [9].

The exoskeletons are capable of recording all arm and hand movements and to replicate forces on the arm and the hand. By moving the control points (see Fig. 2 for the grids of control points) of parametric surfaces according to the data coming from the exoskeletons, we are able to draw a virtual hand and arm that replicates and follows the position and the shape (empirically representing the deformation of the skin) of the real hand and arm.

Also we developed a physically based simulation of the virtual environment for replicating by means of the exoskeletons, on the real arm and hand, forces generates by virtual objects (see [7] [8]).

We obtained 12-15 frames per second with the monoprocessor version of the alghoritm on a Silicon Graphics Personal Iris 4D/35TG with some very simply objects and a virtual arm/hand drawn with 40 nonuniform B-splines approximated by 4x4 polygons meshes (see Fig. 3 for 4x4 and 8x8 approximations) for a total of 640 polygons.

With a very complex environment like that showed in Fig. 1 and 40 nonuniform B-splines approximated by 4x4 polygons meshes for a total of 640 polygons we obtained 16-20 frames per second with the multi-precessor version of the alghoritm running on a Silicon Graphics Powers 440VGX (a multi-processor with four MIPS R3000 processors and a fast graphic subsystem).

On the same machine we obtained more than 32 frames per second, by transforming from parametric representation to polygons meshes more than 1300 parametric surfaces per second, for an application requiring a simple virtual scenario.

During the development of the research we make a compared test between our single processor algorithm described in (7) and *nurbssurface()* of SGI Graphics Library. For the test of drawing a virtual hand on an Indigo R4000 150MHz, we obtained the following results:

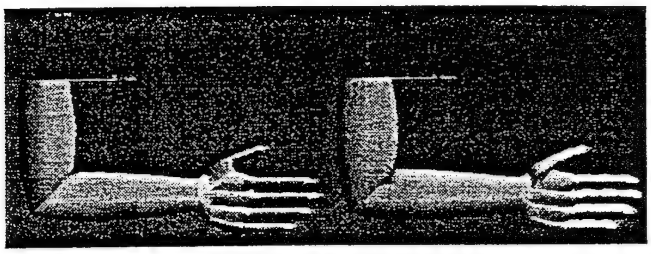


Figure 3: The parametric surfaces approximated with 4x4 polygons (left) and 8x8 polygons (right).

Indigo R4000 150MHz	frame/sec
GL	5.4
New Algorithm	28.9

The GL function achieves a better visual apparency, probably because it uses an adaptative approximation of the parametric surfaces, but our preliminary version of the algorithm is about 6 times faster than the GL equivalent function. With the final version of the algorithm and a multi-processor system, you can achieve an increment of performances of more than one order of magnitude. This is a crucial point with the constraints imposed by a real time application as Virtual Reality.

5 Conclusions

The article we have presented describes a new approach to the design of virtual hand and arm in a Virtual Environment. A complete description of a new mono and multi-processor alghoritm has been presented. Experimental results on a high performance real-time practical application has been described.

Acknowledgments

The research activity described in this article has been developed under the project EP5363 GLAD-IN-ART and Esprit Basic Research 6358 SCATIS.

References

[1] Michael E.Monterson, *"Geometric Modeling,"* John Wiley & Sons, 1985.

[2] Faux, I.D. and Pratt, M.J., *"Computational Geometry for Design and Manufacture"* , Ellis Horwood, Chichester UK, 1979.

[3] Farin, G., *"Curves and Surfaces for Computer Aided Design"*, 2nd edn, Academic Press, Boston, 1990.

[4] David F.Rogers, J.Alan Adams, *"Mathematical Elements for Computer Graphics,"* Mc Graw Hill, 1990.

[5] Alan Watt, Mark Watt, *"Advanced Animation and Rendering Techniques"* , Addison-Wesley, 1992.

[6] M. Bergamasco, B. Allotta, L. Bosio, L. Ferretti, G. Parrini, G.M. Prisco, F. Salsedo, G. Sartini, "An Arm Exoskeleton System for Teleoperation and Virtual Environments Applications," *IEEE International Conference on Robotics and Automation*, San Diego. CA, May 1994.

[7] M. Bergamasco, P. Degl' Innocenti, D. Bucciarelli, G. Rigucci, "Grasping and Moving Objects in Virtual Environments: a preliminary approach towards a relistic behaviour," *IEEE International Workshop on Robot and Human Communication*, ROMAN'94, Nagoya, Japan, July 1994.

[8] M. Bergamasco, P. Degl' Innocenti, D. Bucciarelli, "A Realistic Approach for Grasping and Moving Virtual Objects." *IEEE/RSJ/GI International Conference on Intelligent Robots and Systems 1994*, IROS'94, Monaco, September 1994

[9] M. Bergamasco, F. Salsedo, L. Ferretti, A. Scoglio, G. Piperno, *"Force Feedback Interfaces for Virtual Environments Applications,"* submitted to *MONTPELLIER'95*, Montpellier, France, June 1995.

Evolutionary Art in Virtual Worlds*

Ik Soo Lim†

Institute of Systems Science, National University of Singapore, Heng Mui Keng Terrace, Kent Ridge, Singapore 0511

lis@iss.nus.sg

Abstract

Fourier representations are suggested for shape and texture generation of interactive evolution in virtual environments. With simple input, a user in a virtual environment can achieve complexity (e.g. in shape/texture generation and morphing) fast, free from knowledge of technical details and preprocessing.

Key words: interactive evolution, shape generation, texture generation, morphing, Fourier series.

1 Introduction

Virtual reality(VR) is a method of interacting with a computer-simulated environment. VR is supposed [16, 5] to contain a substantial portion of the following: surround vision, stero cues, viewer-centered perspective, real-time interaction, tactile feedback, *etc.* To give a user the experience of not merely being but acting there, intuitive direct manipulation (for example, interactive volumetric sculpting [7]) is important. Though the idea is simple, making it work in practice is not trivial: the data stream from the sculpting tool may be noisy and the tool has to be an absolute device instead of a relative one for the natural mapping of the physical space of the tool to the screen space representation of it [7]. Besides, significant lag between hand motion and the screen update could be seriously troublesome [13, 21].

Even when this sort of hardware problems are all resolved, the idea of direct manipulation might not be always the best choice for interaction in virtual environments. For complex results under the scheme of direct manupulation, complex input from the user is necessary: say, a virtual sculpture would require very good eye-hand coordination, a great deal of effort and patience. Creating sculptures beyond a certain level of complexity can be all but impossible, especially for a non-artist [1].

Though the results may not be always predictable, the technique of *interactive evolution* [6, 17] lets the user achieve complexity with a minimum of user input and knowledge of details which can augment and enhance the power of direct manipulation.

The next section briefly explains the idea of genetic algorithms, on which interactive evolution is based. Applications of interactive evolution in computer graphics are reviewed in Section 3. Section 4 reviews a use of it in the CAVE virtual environment [4], which had drawbacks in respect of elapsed time. In Section 5 and 6 we describe the po-

*2nd Eurographics Workshop on Virtual Environments, 31 January–1 Feburary 1995, Monte Carlo, Monaco.

†Thanks to T. Poston for helpful discussions.

tential of the Fourier representation for fast generation of shapes and textures, and morphing between them. Discussions and conclusions follow.

2 Genetic Algorithms

Genetic algorithms have been used to solve a wide variety of problems [8]. Used as an optimization technique, genetic algorithms have proven to be an effective way to search extremely large or complex solution spaces. Among such spaces are the vast domains of shape and texture. The search for pleasing combinations of the two cannot be carried out exhaustively, and aesthetic values cannot easily be parametrized [5]. Since genetic algorithms do not rely on problem-specific knowledge, they can be used to discover solutions that would be difficult to find by other methods. The *genotype* is the genetic information that codes for the creation of an individual. The *phenotype* is the individual itself, or the form that results from the development rules and the genotype. The *fitness function* is used to evaluate the relative quality of each phenotype, and the genotypes corresponding to the phenotype judged "best" are used as the basis for the next generation [8, 17, 5]

3 Interactive Evolution

Interactive evolution provides a powerful new technique for enabling human-computer collaboration. It is potentially applicable to a wide variety of search problems, provided the candidate solutions can be *produced quickly* by a computer and *evaluated quickly and easily* by a human [1]. Since humans are often very good and fast at processing and assessing pictures, interactive evolution is particularly well suited to search problems whose candidate solutions can be represented visually.

While traditional genetic algorithms usually use an explicit analytic expression for a fitness function to be evaluated by the computer [8], with interactive evolution the user performs this step based on visual perception [17, 1].

The beauty of interactive evolution is that the user does not have to state or even understand an explicit fitness criterion; the need is only be able to apply it. This feature of interactive evolution is used very effectively by Sims [17] in creating beautiful, abstract color images. An initial population of images, either generated randomly by the computer or input by the user, is displayed on the screen. From the displayed set the user selects one image for mutation or two images for mating: *mutation* is a reproduction technique where a chosen parent's genotype (a Lisp expression) is randomly altered, deriving children which replace the current population. *Mating* or *crossover* is the combination of the two genotypes. Each genotype in the new generation contains a part of both parents' genotypes. Once again the new population (with 'too long to render' types weeded out) replaces the old population. The mating and/or mutation operations are applied to the selected images to produce a new set of progeny images that supply the input for the next round of user selection. This process is repeated multiple times to evolve an image of interest to the user. Evolved images may be saved and later recalled for mating with other evolved images.

There are other noticeable applications of interactive evolution [1, 2, 20] since the inspiring work of Richard Dawkins [6].

4 Related Work in a Virtual Environment

The technique of interactive evolution was also employed in a virtual environment, CAVE [4], to generate shape and sound [5]. Besides being manipulated (e.g. rotated and moved), objects can be mutated or mated to another object to generate new objects of

different shapes.

In this scheme the objects are isosurfaces $\{E(x,y,z) = T\}$, where the genotype specifies the mathematical function E, which can become quite complex. Since the surface extraction is done by the CPU-intensive Marching Cubes Algorithm (with evaluation of E at all cubic grid points), it is slow; and the large number of triangles typical of a Marching Cubes surface gives long rendering times.

This could be serious drawbacks in an application of interactive evolution because *the human's speed and patience are limiting factors* [1]. In the next section, we suggest Fourier representation which will allow *fast* surface generation, with no restriction on the class of surfaces representable.

5 Fourier Surfaces

Fourier representations express a function in terms of an orthonormal basis. One of the motivations for a basis representation is that it allows us to express any obejct as weighted sum of a set of known functions.

A surface in 3D can be described explicitly by three functions of two surface parameters:

$$\mathbf{x}(u,v) = (x(u,v), y(u,v), z(u,v))$$

where u and v vary over the surfaces and x, y and z are the associated Cartesian coordinates. This representation imposes no restriction on the class of surfaces representable. In order to represent surfaces, a basis for functions of two variables is needed. The following can be used [3, 19]:

$$\begin{aligned} cc_{l,m} &= \cos(2\pi lu)\cos(2\pi mv) \\ sc_{l,m} &= \sin(2\pi lu)\cos(2\pi mv) \\ cs_{l,m} &= \cos(2\pi lu)\sin(2\pi mv) \\ ss_{l,m} &= \sin(2\pi lu)\sin(2\pi mv) \end{aligned}$$

where $l, m = 0, 1, 2, \cdots$

A function is then represented by:

$$f = \sum_{l=0}^{K}\sum_{m=0}^{K} \lambda_{l,m} \quad (a_{l,m}cc_{l,m} + b_{l,m}sc_{l,m} + c_{l,m}cs_{l,m} + d_{l,m}ss_{l,m})$$

where

$$\lambda_{l,m} = \begin{cases} 1 & : \quad l = 0,\ m = 0 \\ 2 & : \quad l = 0,\ m > 0 \quad \text{or} \quad l > 0,\ m = 0 \\ 4 & : \quad l > 0,\ m > 0 \end{cases}$$

truncating the series at K. There are three sets of parameters corresponding to the three coordinate functions, a_x, b_x, c_x, d_x, a_y, b_y, c_y, d_y, a_z, b_z, c_z, d_z: a genotype consists of these parameters. For smooth surfaces, a few dozen of low frequency components will be enough since high frequency components, roughly speaking, correspond to wiggles in surfaces.

These surfaces are expressed by parametric equations, so no process of polygonalizing isosurfaces of an implicit function is necessary. The parametric equations are defined on a 2-D region while the implicit function is on a 3-D region: the function evaluation has complexity of $O(n^2)$ for the one, while $O(n^3)$ for the other. Precomputing of the basis functions at mesh points is also possible, replacing trig function evaluation by lookup and simple arithmetic: it takes 5 seconds to generate each shape in **Fig 1**, **Fig 2** and **Fig3** using a machine of SiliconGraphics IRIS INDIGO (32×32 mesh points are used and the Fourier series are truncated at $K = 8$.) It should be noted that the elapsed time is independent of complexity of the geometry generated.

6 Fourier Textures

Fourier representation could also be used for synthesizing textures. Instead of a Cartesian coordinate, $T(u,v)$ is defined as two-dimensional greyscale texture, where u and

v are coordinates in texture space. If a color texture is to be defined, three functions, $R(u,v)$, $G(u,v)$ and $B(u,v)$ have to be maintained for the color components. Any arbitrary texure can be described with this Fourier representation: the Fourier basis is *complete.*

As in the interactive evolution for the shape generation, Fourier coefficients become genotypes, again. It should be noted a contrast between this and that of Sims [17]: the motivation of using Lisp expressions in genotype and evolving complex functions is to surpass limitation on the set of possible phenotypes (*i.e.*, textures) if genotypes consist of a fixed number of parameters and fixed expression rules [17]. But there is no limitation with Fourier representation; any texture represented with a complex function consisting of Lisp expressions can also be arbitrarily well approximated by a Fourier description.

Sims uses a parallel supercomputer to render the images at interactive speed [17] because a genotype for an image can be a function which has evolved to be increasingly complex (*i.e.*, a sequence of image-processing functions). In general, therefore, as images evolve, more time has to be spent to render the images of increasingly complex functions. However, in the case of the Fourier textures, *constant* time rendering is possible if K is fixed.

7 Fourier Morphing

Morphing techiniques for transforming images have demonstrated remarkable results and have achieved widespread use [11]. In this work, however, we rely on *interpolating* the Fourier coefficients of the shape pairs [9] rather than establishing a mapping between the vertices and edges of the respective objetcs and introducing new vertices and edges for making a one-to-one correspondence/connectivity relationships between vertices [15]. Besides no intervention of users nor any preprocessing, the morphing itself can be done fast: rendering a Fourier surface can be done fast as decribed in the previous section and getting intermediate shapes only requires interpolating the few dozen of the Fourier coefficients.

The task is easy even when changing smoothly between widely varying topologies. It should be noted a constrast, especially in terms of elapsed time perspective, between this work and Fourier *Volume* Morphing [9], the latter of which has to spend time of $O(n^3 \log n)$ in the Fourier transforms besides the time of $O(n^3)$ for the operations involved in blending datasets or rendering an image from a dataset of size $n \times n \times n$: the Fourier coefficients, in our case, are already available at the step of the shape synthesis so that fast morphing is possible. Similarly, morphing between textures could be done fast by interpolating the Fourier coefficients of the textures.

8 Conclusions and Further Work

We suggested Fourier representations for surface/texture generation and morphing between them. Due to the parametric/explicit expressions and precomputation of basis functions, these can be done very *fast* compared to the previous work where supercomputers are being used or explored to be used. We are exploring building a Virtual Art Gallery where users can not only watch art exhitted, but also make their own of complexity without complications from interaction/interface problems.

References

[1] E. Baker and M. Seltzer, Evolving Line Drawings, in the *Proceedings of Graphics Interface '94*, pp. 91–100, 1994.

[2] C. Caldwell and V. S. Johnston, Tracking a Criminal Suspect Through Face Space with a Genetic Algorihm, in the *Proceedings of the Fourth International Conference on Genetic Algorihms*, pp. 461–421, 1991.

[3] M. Cartwright, *Fourier Methods for mathematicians, scientists and engineers*, Ellis Horwood, 1990.

[4] C. Cruz-Neira, D. J. Sandin and T. A. DeFanti, Surrounded-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE, in the *Proceedings of SIGGRAPH '93*, pp.135–142, 1993.

[5] S. Das, T. Franguiadakis, M. Papka, T. A. DeFanti and D. J. Sandin, A Genetic Programming Application in Virtual Reality, in the *Proceedings of the Third IEEE Conference on Fuzzy Systems*, pp.1985–1989, 1994.

[6] R. Dawkins, *The Blind Watchmaker*, W. W. Norton and Company, New York, London, 1987.

[7] T. A. Galyean, Sculpting: An Interactive Volumetric Modeling Technique, in the *Proceedings of SIGGRAPH 91*, pp.267–274, 1991.

[8] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*, Addison-Wesley, 1989.

[9] J. F. Hughes, Scheduled Fourier Volume Morphing, in the *Proceedings of SIGGRAPH 92*, pp.43–46, 1992.

[10] K. Kameryama and O. Ohtomi, A Shape Modeling System with a Volume Scanning Display and Multisensory Input Device, *Presence*, **2**, pp.104–111, 1993.

[11] J. R. Kent, W. E. Carlson and R. E. Parent, Shape Transformation for Polyhedral Objects, in the *Proceedings of SIGGRAPH 92*, pp.47–54, 1992.

[12] J. R. Koza, *Genetic Programming*, MIT Press, Cambridge, MA, 1992.

[13] J. Liang, C. Shaw and M. Green, On Temporal-Spatial Realism in the Virtual Reality Environment, in the Proceedings of ACM UIST '91 Conference, pp.19–25, 1991.

[14] W. E. Lorensen and H. E. Cline, Marching Cubes: A High Resolution 3D Surface Construction Algorithm, in the *Proceedings of SIGGRAPH 87*, pp.83–91, 1987.

[15] R. E. Parent, Shape Transformation by Boundary Representation Interpolation: a Recursive Approach to Establishing Face Correspondences, *Journal of Visualization and Computer Animation*, **3**, pp.219–239, 1992.

[16] D. J. Sandin, Virtual Reality Technologies: Head Mounted Displays, Booms, Monitor and Projection Based Systems, in notes from course #23, *Applied Virtual Reality*, SIGGRAPH 1993.

[17] K. Sims, Artificial Evolution for Computer Graphics, in the *Proceedings of SIGGRAPH 91*, pp.319–328, 1991.

[18] J. R. Smith, Designing Biomorphs with an Interactive Genetic Algorithm, in the *Proceedings of the Fourth International Conference on Genetic Algorihms*, pp.535–538, 1991.

[19] L. H. Staib and J. S. Duncan, Deformable Fourier models for surface finding in 3D images, in *SPIE Vol. 1808 Visualization in Biomedical Computing 1992*, pp.90–104, 1992.

[20] S. Todd, W. Latham and P. Hughes, Computer Sculpture Design and Animation, *Journal of Visualization and Computer Animation*, **2**, pp.98–105, 1991.

[21] C. Ware and R. Balakrishnan, Target Acquisition in Fish Tank VR: The Effect of Lag and Frame Rate, in the *Proceeding of Graphics Interface '94*, pp.1–7, 1994.

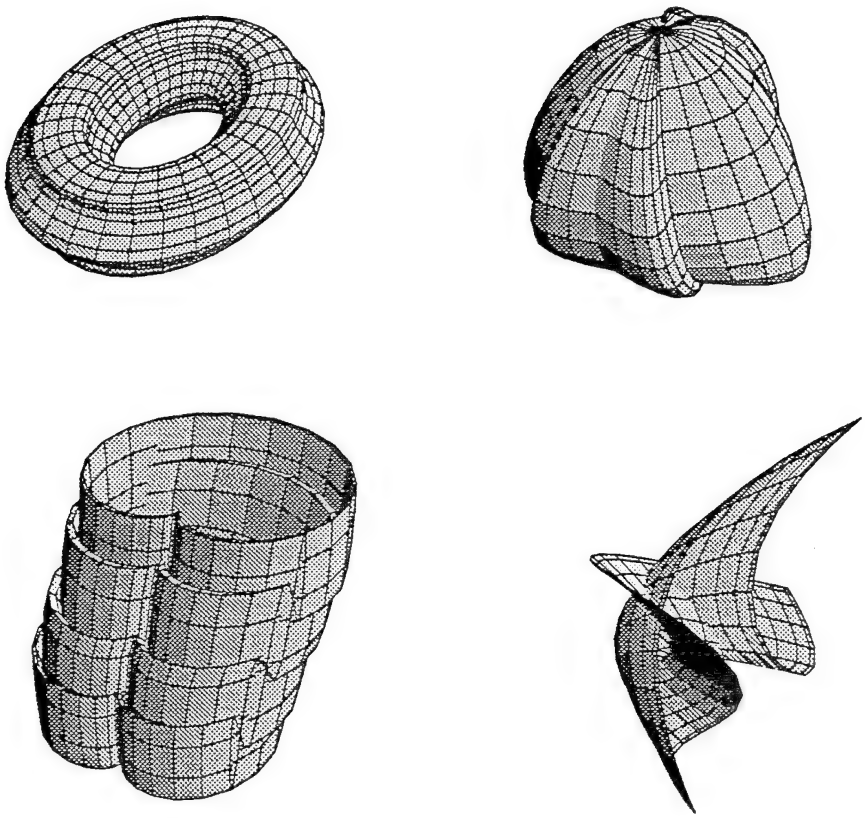


Fig 1, Different types of surfaces representable by the Fourier representation:
Torus(Upper Left), Tube(Lower Lefttt), Open Surface(Lower Right), Closed Surface(Upper Right)

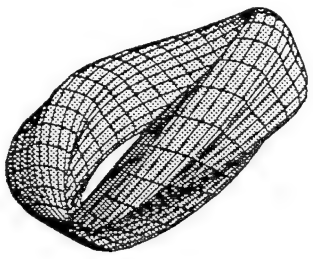


Fig 2(a) Mating. The parents are the torus and the closed surface in Fig 1.

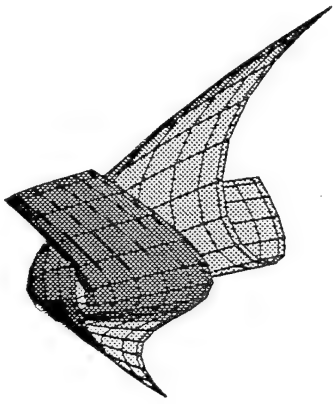


Fig 2(b) Mutation. The parent is the open surface in Fig 1.

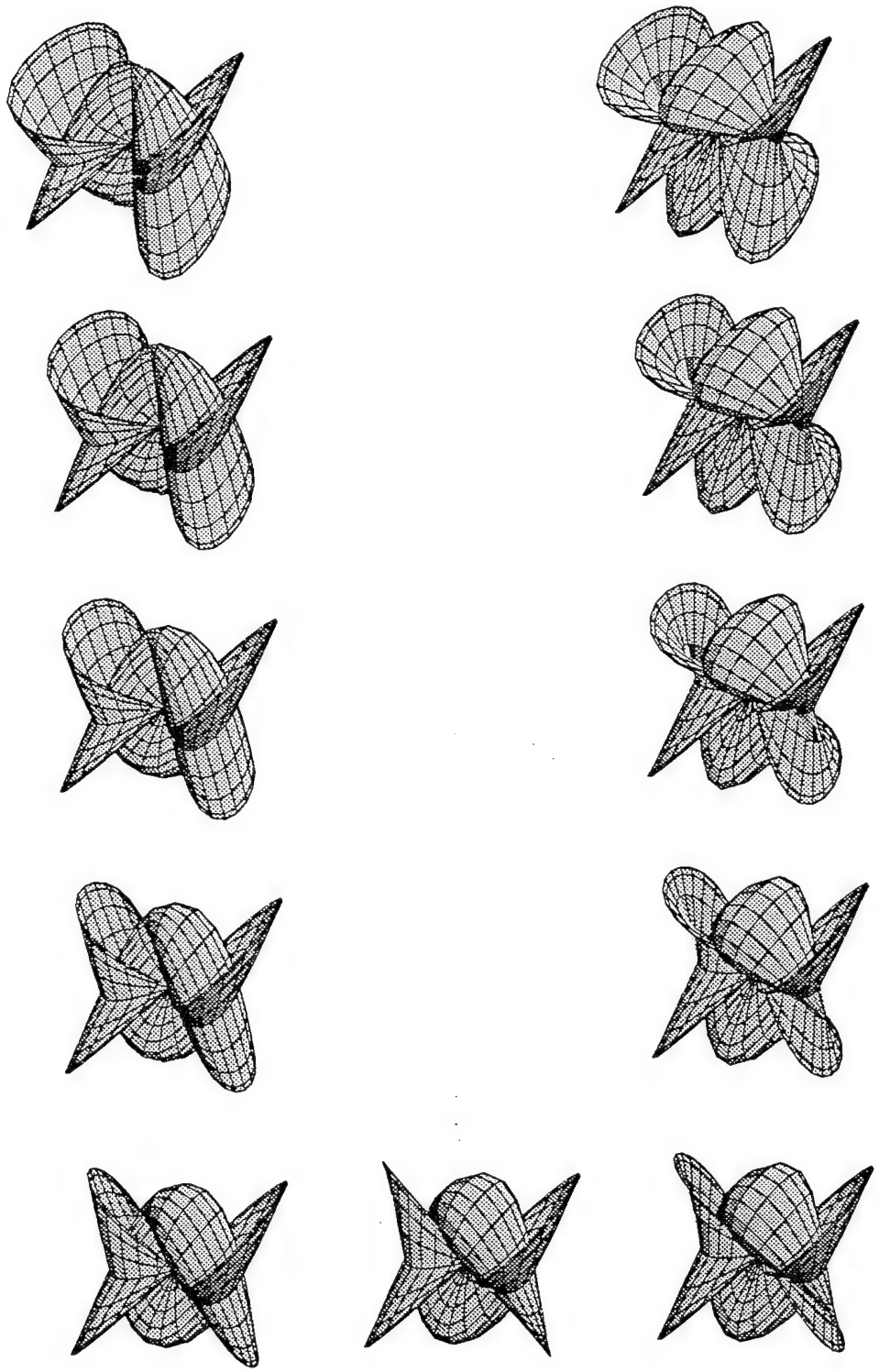


Fig 3 Fourier Morphing. Start from the top right and follow in the clockwise direction and finish at the top left. Or the other way.

Generating Multiple Levels of Detail from Polygonal Geometry Models

G. Schaufler and W. Stürzlinger

GUP Linz
Johannes Kepler University Linz
Altenbergerstrasse 69, A-4040 Linz, Austria/Europe
[schaufler | stuerzlinger]@gup.uni-linz.ac.at
Tel.: +43 732 2468 9228

Abstract

This paper presents a new method for solving the following problem: Given a polygonal model of some geometric object generate several more and more approximative representations of this object containing less and less polygons. The idea behind the method is that small detail in the model is represented by many spatially close points. A hierarchical clustering algorithm is used to generate a hierarchy of clusters from the vertices of the object's polygons. The coarser the approximation the more points are found to lie within one cluster of points. Each cluster is replaced by one representative point and polygons are reconstructed from these points. A static detail elision algorithm was implemented to prove the practicability of the method. This paper shows examples of approximations generated from different geometry models, pictures of scenes rendered by a detail elision algorithm and timings of the method at work.

1 Introduction

In several papers on recent developments in computer graphics the presented algorithms rely on the availability of several approximations of decreasing complexity to the polygonal representation of one geometric object (referred to as levels of detail - LODs - throughout this paper). Primarily for performance reasons these algorithms choose one of the approximative representations of the objects in the course of their work thereby trading quality for speed. Examples are the visualization of complex virtual environments [Funk93], 3D graphics toolkits [Rolf94] or indirect illumination calculations [Rush93]. In visualization of virtual environments LODs allow to retain a constant frame rate during the navigation through the environment by adapting the detail of the visible objects to the complexity of the visible part of the scene and the graphics performance of the used hardware. Indirect illumination calculations attain performance gains through substituting different LODs while calculating the energy exchange between two objects.

The authors of these papers do not mention how to automatically obtain the different LODs. Either they use very coarse approximations such as bounding volumes or they model each LOD by hand thereby multiplying the effort for generating the geometry database.

In the field of surface reconstruction from laser range device data algorithms have been developed which allow to decimate the number of triangles used to represent the original surface. However, these algorithms are very costly as far as computational complexity is concerned and are less suitable for geometry models of CAD software [Hopp94]. Other methods filter large triangle or polygon meshes with the aim to retain the detail of the digitizing process but do not generate several LODs (see e.g. [DeHa91],[Turk92],[Schr92]). The presented method fills the gap between the computationally and algorithmically complex surface reconstruction from laser range data by a user selectable number of triangles and the generation of models of varying complexity by hand. It automatically generates different LODs from CAD geometry models. The original model is referred to as the level 0. Coarser LODs are referred to as level 1, level 2 and so on.

The goal in designing the method was execution speed while keeping a close resemblance to the original model. In coarser LODs the number of polygons must decrease significantly. Small details of the model can be left out but the overall structure of the object should stay the same with as little polygons as possible. It is important to keep the silhouette of the object to minimize annoying artifacts during blending between different LODs and to keep the error low which is introduced into the algorithm making use of the LODs.

The method uses a hierarchical clustering algorithm to perform the task of removing the detail from the models and generating coarser approximations to the models. In the first stage the clustering algorithm generates a hierarchy of clusters from the points in the model. Different algorithms exist to generate such a hierarchy - the "centroid" or "unweighted group pair" method [Snea73] is used in the current implementation. Depending on the desired degree of approximation the method traverses the calculated cluster tree to a specific depth and either keeps the geometry model or replaces parts by coarser approximations.

The current implementation of the proposed method is suitable to calculate LODs from polygonal objects containing several 1000 data points. In an interactive viewer the generated LODs are used to render environments of geometric objects at interactive frame rates. The viewer chooses a LOD for each visible object depending on the size of its on screen projection without introducing highly noticeable artifacts into the rendered views. The program allows navigation through the environment as well as a close inspection of the objects' LODs generated by the method.

2 Previous work

Hoppe et al. [Hopp94] have presented a method to solve the following problem: Starting from a set of three dimensional data points and an initial triangular mesh they produce a mesh of the same topology which fits the data well and has a smaller number of vertices. They achieve this by minimizing an energy function which explicitly models the competing desires of conciseness of representation and fidelity to the data by using three basic mesh transformations to vary the structure of the triangle mesh in an outer optimization loop. For each triangle mesh found they optimize the energy function by slight changes to the vertex positions of the triangle mesh. The method is fairly complex and demanding as far as computational and implementational efforts are concerned.

DeHaemer et al. [DeHa91] have presented two methods for approximating or simplifying meshes of quadrilaterals topologically equivalent to regular grids. The first method applies adaptive subdivision to trial polygons which are recursively divided into smaller polygons until some fitting criterion is met. The second method starts from one of the polygons in the mesh and tries to grow it by combining it with one of its neighbours into one bigger polygon. Growing the polygon stops when the fitting criterion is violated. These methods work well for large regular meshes and achieves reductions in polygon numbers down to 10% for sufficiently smooth surfaces.

Turk's re-tiling method [Turk92] is best suited for polygonal meshes which represent curved surfaces. It generates an immediate model containing both the vertices from the original model and new points which are to become the vertices of the re-tiled surface. The new model is created by removing each original vertex and locally re-triangulating the surface in a way which matches the local connectivity of the initial surface. It is worth mentioning that models containing nested levels of vertex densities can be generated and that smooth interpolation between these levels is possible.

Schroeder et al. [Schr92] deal with decimation of triangle meshes in the following way: In multiple passes over an existing triangle mesh local geometry and topology is used to remove vertices which pass a distance or angle criterion. The holes left by the vertex removal are patched using a local triangulation process. Using their approach Schroeder et al. successfully decimate triangle meshes generated with the marching cube algorithm down to 10%.

All these approaches have in common that they start out from polygon meshes which contain a lot of redundancy: they exploit the fact that in most areas of the surfaces curvature is low and vertices or triangles can be left out or combined without losing features of the surface. Vertices in such areas fullfil the preconditions under which simplifications to the mesh are made. However, such preconditions for simplification are rarely met in human modelled CAD objects where flat surfaces are constructed from large polygons. They cannot be further simplified with these strategies. Moreover the approaches are quite complex to implement and computationally intensive as they all involve re-triangulation and multiple passes over the polygon mesh.

It has been pointed out by a reviewer that the proposed method is similar to the method introduced by Rossignac et al. [Ross92]. Rossignac uses a regular grid to find points in close proximity which is disadvantageous in the case of greatly differing object sizes. The method proposed in this paper uses a hierarchical object-adaptive clustering scheme instead which can deal with such differences and does not introduce any grid size into the generated models. The method does not require the polygon mesh to contain a lot of redundancy nor does it require the mesh to be of any predetermined type or topology. In fact it is well suited to continue from where the above algorithms finish. It works on polygonal objects of several 1000 vertices as in hand modelled CAD objects and explicitly generates several LODs from these objects in one pass over the object's geometry while retaining the overall shape and size of the objects. It is designed to be fast and efficient on such models and is well suited to generate the LODs while the object database is loaded into memory.

3 Overview of the Approach

This approach implements a reasonable fast method to generate several LODs from polygonal object models. This papers terminology refers to the original model as level 0. The algorithm was not required to exactly keep the topology of the geometry as did the algorithms mentioned in the former section. Nevertheless the generated LODs resemble the original model closely though with less and less polygons. The coarsest LOD should not contain more than a few dozen polygons for an original object consisting of several 1000 polygons.

The algorithm can be described as follows:

- Apply a hierarchical clustering algorithm to the vertices of the object model to produce a tree of clusters.
- For each LOD generate a new (less complex) object model using cluster representatives instead of original polygon vertices.
- Remove multiply occurring, redundant primitives from each LOD.

In the second stage a layer in the cluster tree is determined which describes the way the LOD approximates the original model. Using the clusters of this layer, each polygon has its vertices replaced by the representative of the cluster it belongs to. This may leave the number of vertices unchanged (if all points fall into different clusters), the number of vertices may be reduced, the polygon may collapse into a linestroke or its new representation may be a single point. In this way formerly unconnected parts of the model may eventually become connected when separated points of different polygons fall into one cluster and are, therefore, mapped onto one cluster representative. However, as the clustering algorithm only clusters spatially close points the overall appearance of the object remains the same depending on the degree of approximation desired in the current LOD. In particular polygons become bigger if their points are moved apart through clustering. Therefore, the surface of the object will never be torn apart.

4 Hierarchical Clustering

Clustering algorithms work on a set of objects with associated description vectors, i.e. a set of points in multidimensional space. A dissimilarity measure is defined on these which is a positive semidefinite symmetric mapping of pairs of points and/or clusters of points onto the real numbers (i.e. $d(i,j) \geq 0$ and $d(i,j) = d(j,i)$ for points or clusters i and j). Often (as in this case) the stronger distance is used, where in addition the triangular inequality is satisfied (i.e. $d(i,j) \leq d(i,k) + d(k,j)$) as in the Euclidean distance. The general algorithm for hierarchical clustering may be described as follows (although very different algorithms exist for the same hierarchical clustering method [Murt83]):

Step 1 Examine all inter-point dissimilarities, and form a cluster from the two closest points.

Step 2 Replace the two points clustered by a representative point.

Step 3 Return to Step 1 treating constructed clusters the same as remaining points until only one cluster remains.

Prior to Step 1 each point forms a cluster of its own and serves as the representative for itself. The aim of the algorithm is to build a hierarchical tree of clusters where the initial points form the leaves of the tree. The root of the tree is the cluster containing all points.

Whenever two points (or clusters) are joined into a new cluster in Step 2, a new node is created in the cluster tree having the two clusters i and j as the two subtrees. For the new cluster a representative is chosen and a dissimilarity measure is calculated describing the dissimilarity of the points in the cluster. The representative g of the new cluster is calculated from

$$g = \frac{|i|\, g_i + |j|\, g_j}{|i| + |j|}. \qquad |i| = \text{number of points in cluster i.}$$

The dissimilarity measure d equals the distance of the two joined clusters i and j:

$$d = d(i,j) = \|g_i - g_j\|.$$

The run-time complexity of this approach can be analyzed as follows: Step 1 is the most complex in the above algorithm ($O(N^2)$ in a naive approach). The time required to search the closest pair of points can be decreased by storing the nearest neighbours for each point or cluster. Therefore, a BSP-tree is built from the points at the cost of $O(N\log N)$ to facilitate the initialization of the nearest neighbour pairs (at the equal cost of $O(N\log N)$).

Each time Step 2 is performed the nearest neighbour data is updated at the cost of $O(N)$ on the average yielding a total complexity of the algorithm of $O(N^2)$. As stated in the paper of Murtagh [Murt83] hierarchical clustering can be implemented with a complexity of less than $O(N^2)$, so a

future implementation should incorporate such optimizations or one of those clustering algorithms to make the method suitable for dealing with more complex geometry models.

5 Finding the Approximative Polygonal Model

The tree of clusters generated by the hierarchical clustering algorithm is used as the input to the next stage of the method - the automatic generation of several LODs. Starting from the highest LOD (coarse approximation) the algorithm proceeds towards level 0 (the original model). For each level to generate a size of object detail is determined which can be ignored in this level. The found size will be used to choose nodes in the cluster tree which have dissimilarity measures (i.e. cluster distances) of similar magnitude. 1/8th of the room diagonal of the object's bounding box works well in most cases for the coarsest approximation.

Next a descend into the cluster tree is made starting from the root until a node is found the dissimilarity measure of which is smaller than the neglectable detail size of the current LOD. For each such cluster a representative is calculated and all the points in the cluster are replaced by this representative. The point which is furthest away from the object's centre is taken as the cluster representative. Although other methods are possible (see "Results and Timings" on page 7) this one delivered the most satisfying results. It keeps the overall shape and size of the objects and has the additional advantage that no new vertices need to be introduced into the object's model. Using the cluster average might result in better looking shapes of the LOD but tends to produce LODs which are increasingly smaller compared to the original object and results in annoying effects during rendering.

From the found layer of clusters in the cluster tree a new geometric object with the same amount of (possibly degenerate) polygons is calculated, only the point coordinates of some of the polygons vertices change. Now each polygon is examined to find out whether it is still a valid polygon or if it has collapsed into a line stroke or a point. Moreover, polygons with more than 3 vertices might need to be triangulated as their vertices are not plane in general. Polygon normals are calculated anew for each LOD. In models of tessellated curved surfaces the normals may be interpolated at the vertices if desired.

6 Cleaning up the New Model

As polygons may collapse into line strokes and points, care must be taken not to incorporate unnecessary primitives into the LODs. For example lines or points which appear as an edge or vertex of a polygon in the current LOD can be discarded. Even certain polygons can be left out.

In a second pass over the new approximated geometry those primitives in the LODs are identified which can be removed without changing the look of the LOD. All points which occur as a vertex of a valid polygon or of a valid line are flagged and all lines which occur as an edge of a valid polygon, too. Assuming that the models are composed of closed polyhedra (as is usually the case) - all pairs of polygons which contain the same vertices in reversed sequence and all polygons which contain the vertices of another valid polygon in the same orientation can be flagged as well.

For the sake of speed these flagged primitives can be removed from the LOD geometry. However, if it is required to retain the relationship between the primitives in the LODs and the original geometry model, the flagged primitives can be retained and need not be processed.

The next (finer) LOD is generated in the same way, only the size of neglectable object details is decreased by a certain factor (in the current implementation the factor is 2, which proved to work well). Therefore, a lower layer is chosen in the cluster tree where clusters are smaller and represent

smaller features. As a result less detail of the original model is left out because less points fall into one cluster.

Different LODs generated by the method are shown in Fig. 1. Top left shows the original model, the other three groups of pictures show higher LODs of the object in a close-up (big pictures) and magnified versions of both the original model (small left) and the LOD (small right) from a distance for which this LOD is appropriate.

7 A Viewing System Based on Levels of Detail

A viewing system using the LODs was implemented both to visually control the results of the method and to demonstrate the usability of the LODs in rendering of complex virtual environments. Generally speaking the incorporation of LODs into the viewer resulted in sufficient speedup for near interactive frame rates (5-10 Hz) on a graphics system with a peak performance of 25K polygons per second.

Prior to drawing the scene the system determines the visible objects by classifying their bounding boxes to the viewing frustum. For the potentially visible objects the length of the projection of the bounding box diagonal is calculated to give a measure at which LOD the object should be rendered. Then all the polygons facing the viewer are rendered using the Z-buffer algorithm.

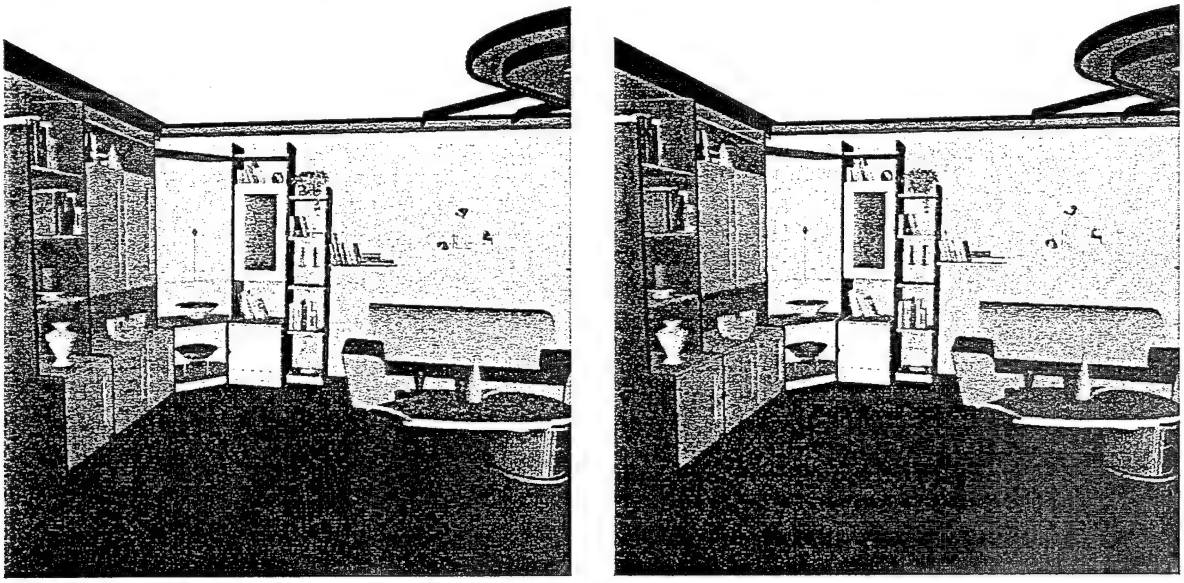


Fig. 2 Living Room left with, right without LODs (left 6810, right 13847 polygons)

8 Results and Timings

The proposed method for the automatic generation of LODs works well for polygonal object models of some 1000 vertices. Models can contain any kind of polygons (convex, concave, general) as long as the triangulation algorithm for approximated polygons can handle them. However, convex polygons are preferable for high rendering speed. Lines and points are legal primitives as well. The objects' meshes should be closed although a violation of this requirement does not prevent the algorithm from being used on the model.

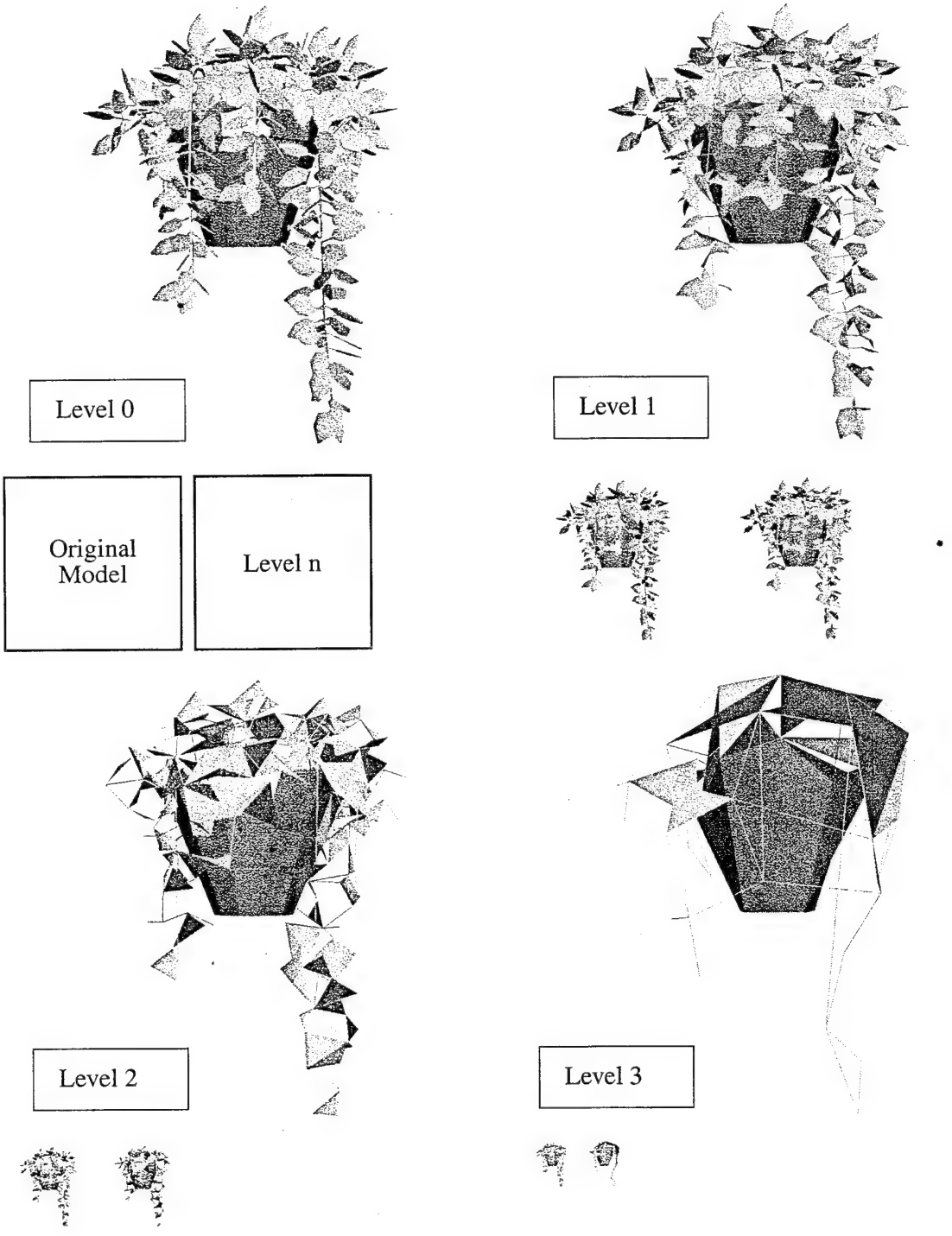


Fig. 1 Levels of Detail of Plant (6064, 3674, 1225, 339 polygons/lines)

It is possible to vary the method in several ways to better match the requirements of the application

for which the LODs are generated. First the number of LODs generated for each object can be adapted by changing the factor by which the dissimilarity measure is updated from level to level. Second the selection of the representative for each cluster can be chosen from a variety of possibilities (cluster average, point in cluster closest to cluster average, point in cluster furthest from object centre, random point in cluster...). Third the primitives generated by the method can be made to include points, lines or polygons or any combination. Fourth the degenerated polygons can be left within the LODs to keep the relationship between the primitives in the original model and the LODs. This is useful for transferring information from the primitives of one LOD to the other (e.g. color, texture, ...).

The method was applied to a wide variety of objects achieving acceptable results as far as speed and quality of the approximation are concerned. The resulting LODs are not easily distinguished from the original models when viewed from increasing distances but contain far less polygons than the original models.

Fig. 3 shows some more examples of objects and their LODs from a close as well as from a suitable viewpoint, i.e. an appropriate distance for the use of the respective LOD. The statistics of the algorithm are summarized in Table 1 for the objects in Fig. 1 and Fig. 3. They were obtained on a Mips R3000 CPU.

Model	Level 0 Polys/Points	Level 1 (polygons)	Level 2 (polygons)	Level 3 (polygons)	Level 4 (polygons)	Time (sec) Total for 4 LOD
Lamp	790 /3084	492 (62%)	210 (27%)	63(8%)	37 (5%)	2.9
Fitting	1581 /5301	1325 (43%)	833 (27%)	225(7%)	55 (2%)	4.2
Shelves	1783 /6628	740 (42%)	472 (26%)	454(25%)	231 (13%)	3.0
Plant	6064 /25926	3674 (61%)	1225 (20%)	339(6%)	57 (1%)	57.0

Table 1 Statistics for the new method

9 Future Work

As already mentioned briefly we are investigating at two areas of further research. First we want to further speed up the hierarchical clustering algorithm by making best use of the optimizations described by Murtagh [Murt83]. This work will decrease the order of complexity of the clustering algorithm below $O(n^2)$ allowing to deal with more complex object models in less time.

Second we want to build a new mode into our viewer which guarantees a constant frame rate through the method described by Funkhouser et al. [Funk93]. We hope to outdo their results in variations in the frame rate as our object models are well structured and probably allow faster and better estimation of the rendering complexity of the visible objects.

Further research is needed to increase the flexibility of our method and to further improve the overall look of the LODs. Moreover, we are investigating the applicability of LODs to other application areas.

Fig. 3 Level of detail close and distant

References

[DeHa91] Simplification of Objects Rendered - Polygonal Approximations. *Michael J. DeHaemer, Jr. and Michael J. Zyda.* Computers & Graphics Vol. 15, No.2 pp 175 - 184, 1991.

[Funk93] Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments. *Thomas A. Funkhouser, Carlo H. Séquin.* SIGGRAPH 93, Computer Graphics Proceedings, pp 247-254, 1993.

[Hopp94] Piecewise Smooth Surface Reconstruction. *Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jun, Jon McDonald, Jean Schweitzer, Werner Stuetzle.* SIGGRAPH 94, Computer Graphics Proceedings, pp 295-302, 1994.

[Murt83] A Survey of Recent Advances in Hierarchical Clustering Algorithms. *F. Murtagh.*The Computer Journal, Vol. 26, No. 4, 1983.

[Rolf94] IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics. *John Rohlf, James Helman.* SIGGRAPH 94, Computer Graphics Proceedings, pp 381-394, 1994.

[Ross92] Multi-Resolution 3D Approximations for Rendering Complex Scenes. *J. R. Rossignac, P. Borrel.* Computer Science, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, 2 1992.

[Rush93] Geometric Simplification for Indirect Illumination Calculations. *Holly Rushmeier, Charles Patterson, Aravindan Veerasamy.* Proceedings of Graphics Interface '93, 1993.

[Snea73] Numerical Taxonomy; The Principles and Practice of Numerical Classification. *Peter H. A. Sneath, Robert R. Sokal.* Publisher W. H. Freeman, San Francisco, 1973.

[Turk92] Re-Tiling Polygonal Surfaces. *Greg Turk.* SIGGRAPH 92, Computer Graphics Proceedings, pp 55-64, 1992.

[Schr92] Decimation of Triangle Meshes. *William J. Schroeder, Jonathan A. Zarge, William E. Lorensen.* SIGGRAPH 92, Computer Graphics Proceedings, pp 65-70, 1992.

Searching for Facial Expression by Genetic Algorithm

Heedong Ko[+] Jeong-Hwan Kim[*] Jai-Hie Kim[*]

ko@kistmail.kist.re.kr frog@wagner.kist.re.kr jhkim@bubble.yonsei.ac.kr

[+] Korea Institute of Science and Technology (KIST), CAD/CAM Laboratory

[*] Yonsei University, Department of Electronics

Abstract

FACS(Facial Action Coding System) was proposed by the psychologists, Paul Ekman and Wallace V. Friesen, to describe a facial expression. It says a human facial expression is composed of 46 muscular movements called AUs (Action Units). Here, a mesh model of a human face is defined and each action unit is designed to deform the mesh model according to FACS system. Using the mesh model, one can create a facial expression by asserting a set of action units and their intensity values. In this paper, we propose a method using GA (genetic Algorithm), known wide as a function optimizer, to extract the muscular information of an arbitrary facial expression, where the expression may be input by 3D measuring device or 2D image. By using the precise muscular information, we can construct a more realistic facial expressions and this process contributes toward creating an artificial agent inhabiting the virtual world, a virtual agent.

1. Introduction

We receive the external information through the five senses and most of the information is received by the visual sensing system. There may be many methods to provide visual information and a look of a human face is a good example of the means to express a mood or thought of the person. Many meanings can be transmitted by a facial expression. The look reflects the emotion of an individual, happiness, anger, sadness and many other delicate nuances. If the computer expresses its internal state to us using the look of a human face, we may construct a more familiar interface to us. The machine having a human face ! The environment for us to communicate with machines as if they were our friends ! This is a human-computer interface we are pursuing as a part of the virtual reality (VR) research at Korea Institute of Science and Technology (KIST). As a part of VR research, we are investigating for an artificial agent inhabiting a virtual world. The agent may communicate to us in the external world by human facial expression.

The purpose of this paper is to present an experimental result in finding out the components of a facial expression automatically. Each component is called an action unit, which defines what part of a face changes and is based on the facial action coding system(FACS)[1].

1.1 Facial Action Coding System (FACS)

FACS was suggested by the psychologists, Paul Ekman and Wallace V. Friesen, in order to describe human facial expressions in the 1970's. According to FACS, there are 46 primitive muscular movements in a human face called action units, and a facial expression can be synthesized by the combination of action units. For example, the expression describing happiness is composed of the muscular movements which raise the inner eyebrows, pull the corners of lips, and drop a jaw. Figure 1. shows appearance changes due to the combination of action units and their relative strengths (or intensities), and Table 1. describes the kinds of action units and their functions.

(a) Expressionless face (b) Happiness (c) Sadness (d) Surprise

(b) AU combination : 1(0.4) +12(0.4) +26(0.4) (c) AU combination : 1(0.3) +4(0.6) + 15(0.4)
(d) AU combination : 1(0.7) + 2(0.7) + 5(0.7) + 26(0.7)

Figure 1. Appearance changes due to AU combinations

No.	AU Function
1	Inner Brow Raiser
2	Outer Brow Raiser
4	Brow Lowerer
5	Upper Lid Raiser
6	Cheek Raiser
7	Lid Tightener
8	Lips Toward
9	Nose Wrinkler
10	Upper Lip Raiser
11	Nasolabial Furrow Deepener
12	Lip, Corner Puller
13	Sharp Lip Puller
14	Dimpler
15	Lip Corner Depressor
17	Chin Raiser
18	Lip pucker
19	Tongue Show
20	Lip Stretcher
21	Neck Tightener
22	Lip Funneler
23	Lip Tightener
24	Lip Pressor
25	Lips Parts
26	Jaw Drop
27	Mouth Stretcher
28	Lips Suck
29	Jaw Thrust
30	Jaw Sideways
31	Jaw Clencher
32	Bite
33	Blow
34	Puff
35	Cheek Suck
36	Tongue Bulge
37	Lip Wipe
38	Nostril Dilator
39	Nostril Compressor
41	Lid Droop
42	Slit
43	Eyes Closed
44	Squint
45	Blink
46	Wink

Table 1. Action Units

1.2 Creation of Facial Expressions

If one has the data of one basic expressionless face, he can modify the expression by applying several action units with their intensities - a face is modeled by triangle meshes and applying an action unit means changing the coordinates of the corresponding vertices of the action unit, characterizing the muscular movement and the AU intensity determines the degree of the changes of the coordinates. One can create all facial expressions at his will if he knows the composing elements of the facial expressions. Unfortunately, FACS does not provide a theory how to combine the action units to create a desired facial expression. That is, in order to create a facial expression, one must search for a right combination by trial and error and the search process is very time consuming. For example, if each action unit has 10 intensities, we have to search 10^{46} combinations per facial expression at the worst case. Here, we use a genetic algorithm(GA) to search for the target expression[3].

The search space of the problem we are to solve is very large, so we need the heuristic optimal search technique. Besides, it is a multi-modal function and may be discontinuous[5]. That is, it has so many local peaks in its search region, which makes it difficult to apply a simple method like hill-climbing, which is a direct method of calculus-based search technique and finds the global minimum or maximum only in convex spaces, to locate the global peak. Another widely used method in optimization is the simulated annealing method[6][7], which offers a way to overcome this major drawback of calculus-based method, but the price to pay is a huge computation time and the searching process is basically sequential in nature. To avoid their drawbacks, we apply GA to our problem, which avoids converging into the local maxima by giving multiple initial values and can be implemented easily in parallel.

1.3 Motivation

- Facial mesh data from face sculpturing robot system[2]

At Expo'93, held in Taejon, Korea, the CAD/CAM laboratory of Korea Institute of Science and Technology demonstrated a face sculpturing robot system that obtains the 3D contour of a person's face using a projection type multislit beam topography[2]. The facial features of a person are extracted from the contour of the face by matching the model face in Figure 3-a with the measured data approximately . All the facial features are defined a priori for the model face. Then, the recognized face data is affine transformed at the feature level to show different facial expressions. Each transformation was defined by a combination of action units and their relative strengths a priori by the human user. From the transformed data, a robot program sculpturing the person's face is automatically generated with different cutting conditions for individual features like nose, lips, and others.

The z value of a mesh point is calculated by a linear interpolation of the z values of adjacent grid points determined from the regularly spaced lines parallel to either x or y axis and measured contour data of a face as shown in Figure 2. After the model mesh data for a face is approximately matched with a Z-map data, additional vertices are added to make the surface of a face smooth and expressions look natural as shown in Figure 3-b.

- Facial Expression Composer

The modeling of a human expression was very time consuming and the result was an approximate expression with no individual differences. Initially a facial expression composer was developed to input various action unit combinations and their intensities by a musical chord. This is achieved by mapping each key in the MIDI keyboard to an action unit. Thereby, a MIDI keyboard player can generate an arbitrary facial expression by playing a musical chord. Unfortunately, this method may create an expression with no corresponding human facial expression. As another research issue, this system was extended to animating the change of a facial expression according to a musical chord. A facial expression represents the listener's emotion when listening to a music, changes according to the musical passage. This is the mapping of the emotion to the expression. A facial expression changes according to the musical passage when the mapping between musical attributes like harmony tempos and pitches are mapped to human emotions which are in turn mapped to facial expressions. The number of the expression type used in that research is nine and each type is subdivided into several sub_expressions depending on the intensities of the actions.

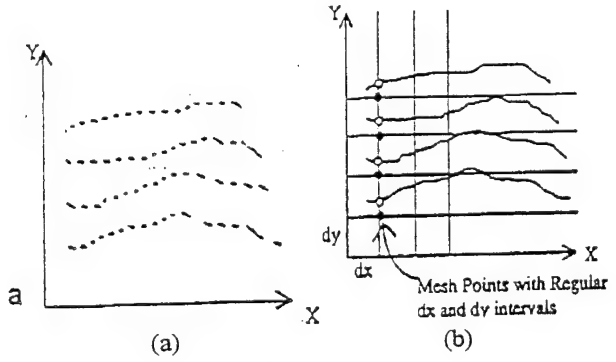


Figure 2.

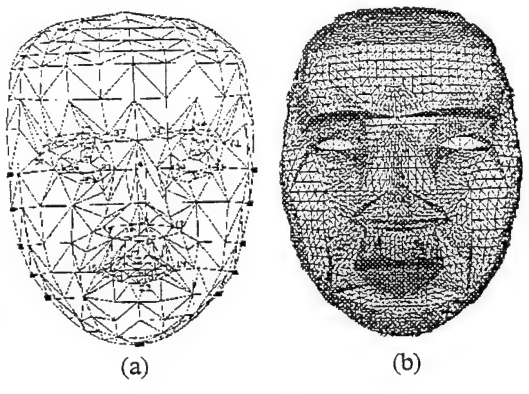


Figure 3.

Figure 2. Conversion of measured data into regular mesh data
Figure 3. Model face with characteristic points and Mesh model of an expressionless face

1.4 Problems

The process mentioned above has limits in several respects. The resultant expressions are not realistic enough because they are generated by a transformation that is defined by a human who laboriously generated the rule by a trial and error for a generic face. When applied to a target face, it can only approximate the target person's facial expression. That is, the synthesized facial expression does not account for individual differences. In addition, the facial expressions are quite diverse, for example, surprise is a distinctive expression from others which may be anger, disgust, happiness, etc., but there is not one unique expression which represent the emotion. We open our eyes and mouths wide to make others notice that we are surprised, but these actions are not all the same. So the number of facial expression we can imagine is enormous and it is not likely that there is a canonical set of primitive facial expressions from which all facial expressions can be generated. It is almost impossible to analyze each individual and search for the right combination of the action units and their intensities, which will compose the expression, one by one depending only on human intuition. Here, we investigate how to find the right combination automatically by applying GA. Genetic algorithm known widely as function optimizer was suggested first by Holland in 1970's[9]. It has a feature of robustness. That is , it is a problem-independent algorithm and so its application is various. The only problem-dependent procedure is an evaluation, which is a major part of problem modeling. Genetic algorithm inspired by an evolution is an adaptive method which can be used to solve the optimization and searching[4][8].

When this effort is successful, an arbitrary human expression can be registered to the system and the system can recognize the person's facial expression among the registered facial expressions

2. Searching for facial expression by Genetic Algorithm

We explain the procedure of applying GA to finding the right combination of action units and their intensities when given the Z map measurement data of an arbitrary facial expression.

An objective facial expression is determined and its vertex data are obtained by measuring the contour of a face. The measuring data of this experiment, which was used by the face sculpturing robot at EXP0'93, is obtained by using a projection type multislit beam topography. Because an action unit acts only on the coordinates of mesh vertices of a face, these measuring data must be converted to the model mesh format. By using characteristic points that are identified on the measuring data as anchors, the model mesh is transformed to fit the measurement data. These intermediary data are used for an evaluation parameters. The combination of action units and their intensities causing minimum difference of the vertex coordinates with a target face ,of which mesh data we already know , is found. The target data obtained by trial and error with human intuition are used for only test, however, if the mesh data of a target face are replaced by ones constructed from the real-measured 3D contour data of an arbitrary facial expression or a model mesh data dynamically matched with 2D picture images, we can obtain the information of muscular movements for arbitrary realistic expressions. So by using GA, we can reduce our problem, synthesizing facial expressions and making canonical sets of facial expressions more realistic, to cost-minimizing heuristic search.

2.1 Two approach methods

The goal of the searching process is to find the parameters causing the global minimum evaluation. The procedure of genetic algorithm, the optimal search technique, was implemented in two different ways. In nature, genetic algorithm is based on a string structure, which represents the chromosome of an object being evolved. The first method models a chromosome with a bit string while the second one implements the combination of action units with an integer array. The first implementation using bit strings is a typical modeling case in the genetic algorithm. Most programs with GA use a binary bit string model because it matches Holland's schema theorem best. However, we also apply another method, an integer model, which is executed with the problem domain reduced and preliminary information of the problem appended.

2.2 Binary bit string model

- Modeling

In the binary bit string model, a chromosome consists of the intensity information of 35 AUs out of 46 possible AUs - only 35 AUs have been implemented - and four bits are aligned to a AU so that 16 intensities can appear. Each bit string is implemented by a gray code not to make a critical disturbance by a bit mutation[3]. In Figure 4, the intensities of AU1 and AU4 are 7 and 4 respectively.

0100	1001	0010	0110.....0110	0101	0001	0101
AU1	AU2	AU3	AU4	AU32	AU33	AU34	AU35

Figure 4. Chromosome model using binary bit string

- Evolutionary process

The GA procedure approximately is divided into four parts, evaluation, selection, crossover and mutation.

.Evaluation function

The evaluation process, as referred earlier, measures the difference between the vertex coordinates of the face caused by the combination of AUs represented by the chromosome string and those of the target face. At the evaluation stage, the scaling is necessary to be suitable for our purpose of continual convergence into the global minimum. That is, the fitness scale is regulated so that the dominant individuals ,which have high fitness values, may not appear in the early stages while they appear in latter stages. If some dominant individuals appear in early stages, the law of natural selection will prevent others from evolving and few dominant individuals will survive. This means the population lose the diversity and the possibility that the searching process pre-converges to a local minima (or maxima) increases. However, if diversity increases too much, in other words, most of all individuals have similar fitness values, the evolutionary process will be excessively time-consuming one and the possibility of oscillation will increase. Therefore, the harmony between two contrary aspects, diversity and convergence, is necessary. In our experiment, the fitness value of an individual is set to the square of a raw fitness and 0.01 is added to an exponent at every generation to enlarge the differences of the fitness values among individuals in a population as shown in Figure 5.

. Selection, Crossover and mutation

The process of reproduction can be divided into selection and crossover. The selection process applies the mixture of the method that always makes the fittest remains in the next generation and fills the population of the next generation with the members of the previous generation according to their expected values of the selection and that of a roulette wheel. We take the multiple point crossover scheme because single point crossover produces insufficient changes for the chromosomes of this experiment, which has no fewer than 144 bits as its genes.

Cossover reveals two aspects. It makes two crossover chromosomes exchange each good nature genes and improves their characters and in the other view point, it gives diversity to the chromosomes by modifying the genes which are the borders during the exchange process of the genes. For our chromosomes, if we apply single crossover scheme, only one among 35 AUs is affected. After many generations , the evaluation converges gradually to some point, but in many times it is trapped to the local minimum or local maximum. In GA, to avoid this situation we must provide the diversity to the population which is supplied, in most cases, by mutation. In our experiment, we have crossover also take the role actively by modifying the number of the points of the crossover dynamically as the distribution of the average fitness of the population is gradually concentrated on some small restricted region. This implementation is opposite to De Jong's opinion[10] - an increased number of crossover points is likely to result in a random shuffle and fewer important schemata can be preserved. However, the

experiment showed it could help to the improvement of a performance, which is shown in Figure 6, for the increased points of crossover operations were used for increasing *diversity* when a population was pre-converging to a local optima.

A mutation rate is adapted to the distribution of the population at each generation. As the normalized average fitness of the population increases, the mutation rate will increase. However, increasing diversity aggravate the converging process, by causing the evaluation function to oscillate. In such cases, the algorithm will be similar to a random search and makes no sense. It is important to harmonize those two aspects of the total process, diversity and stability, to make the evaluation function converge into the global minimum or maximum. In our experiment, the search space is very large and the evaluation function is multi-modal, so it is likely to be trapped to a local optima, so the diversity is emphasized.

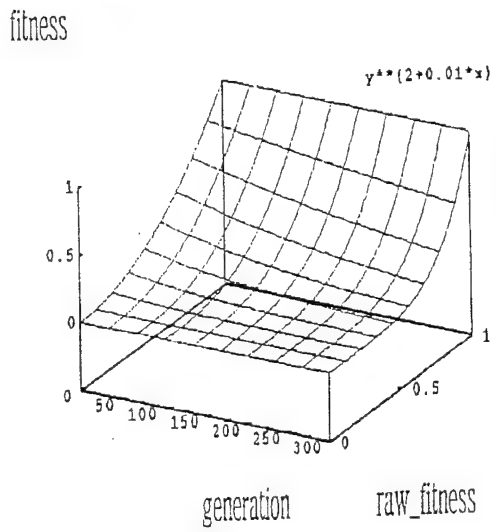


Figure 5. Fitness scaling

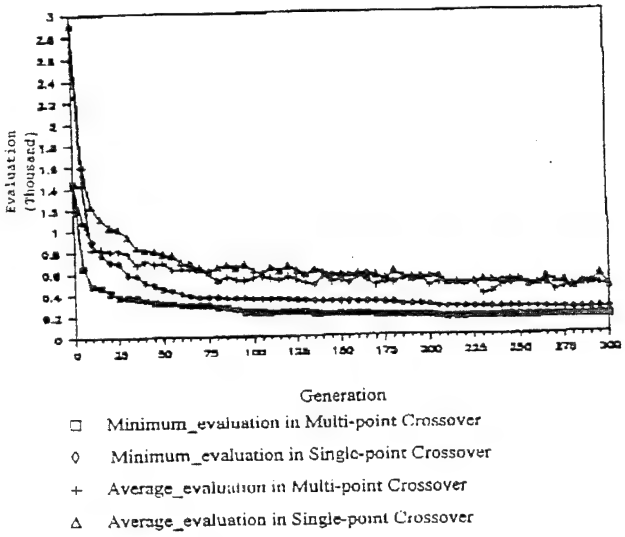
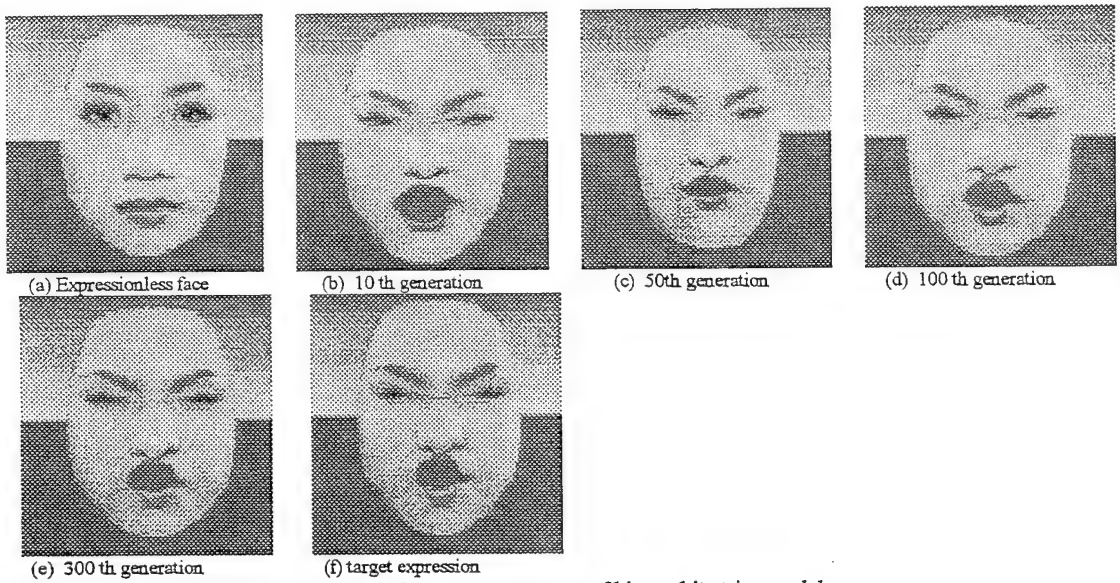


Figure 6. multi-point vs. single point crossover

- **Convergence : experimental result**



(a) Expressionless face (b) 10 th generation (c) 50th generation (d) 100 th generation

(e) 300 th generation (f) target expression

Figure 7. Convergence process of binary bit string model

The individuals of a population gradually approach to an adjacent region in a search space through many generations, and this is called convergence. Figure. 7 shows the convergence process when the size of a population is 300, the possibility of a crossover is 0.7, a mutation rate is the cube of a normalized fitness, and a selection schedule is the mixture of methods using both roulette wheel and expectation value of each individual.

2.3 Chromosome modeling using an integer array

Our second implementation, an integer array model, has a result similar to that of the first except the fast convergence time. It is because we reduced the search space by limiting the number of AUs within a chromosome to 15 because that in most instances the number of the AUs composing a facial expression is less than ten - the search space are reduced from 2^{144} to ${}_{35}C_{15}$. Different from the bit string model, its crossover points are the borders of the AUs, which gives no modification to the AU parameters so that it has to mainly depend on the mutation for its modification. So the mutation rate must be higher than the former's, but this scheme causes the distribution of the evaluation values wide and the average evaluation curves to oscillate more severely. This effect is obvious when comparing the convergence process of average fitness in Figure 8 with that of Figure 6. In addition, in order that a diversity is provided from crossover operation ,too, in case that a chromosome has the same AU as its gene, the intensity of the AU is added or subtracted by the value Difference * Random_gen (MAX_INTENSITY) / MAX_INTENSITY with the probability of 0.5. Figure 8 and Figure 9 show the convergence process of integer array model when the probability of chromosome mutation is 0.3, AU mutation probability is 0.1, the probability of an intensity mutation is 0.2 and other parameters are the same with the former method.

2.4 Difficulties in convergence process

There is a problem we are to solve in both the cases that the numerical evaluation data do not completely converge into the global optimum though the resultant facial expression is almost the same with the target expression when we compare those two with our eyes. It is guessed that the fact is mainly due to the multi-modality of the evaluation function. To solve the problem, it may be useful to apply a hybrid method of the genetic algorithm with the local search techniques such as hill-climbing, and use other strategies, for example adaptation of a mutation probability to a given environment [9]or sequential niche method [5] , etc. Many papers that apply GA's parallelism to the sequential methods such as simulated annealing[6][7] have been published and those concepts will also be helpful to the oscillation problem.

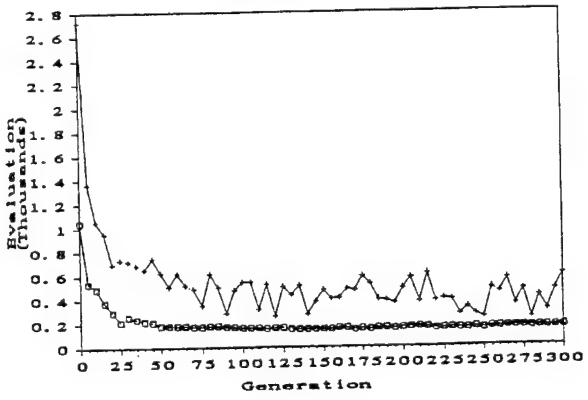


Figure 8. Evolutionary process

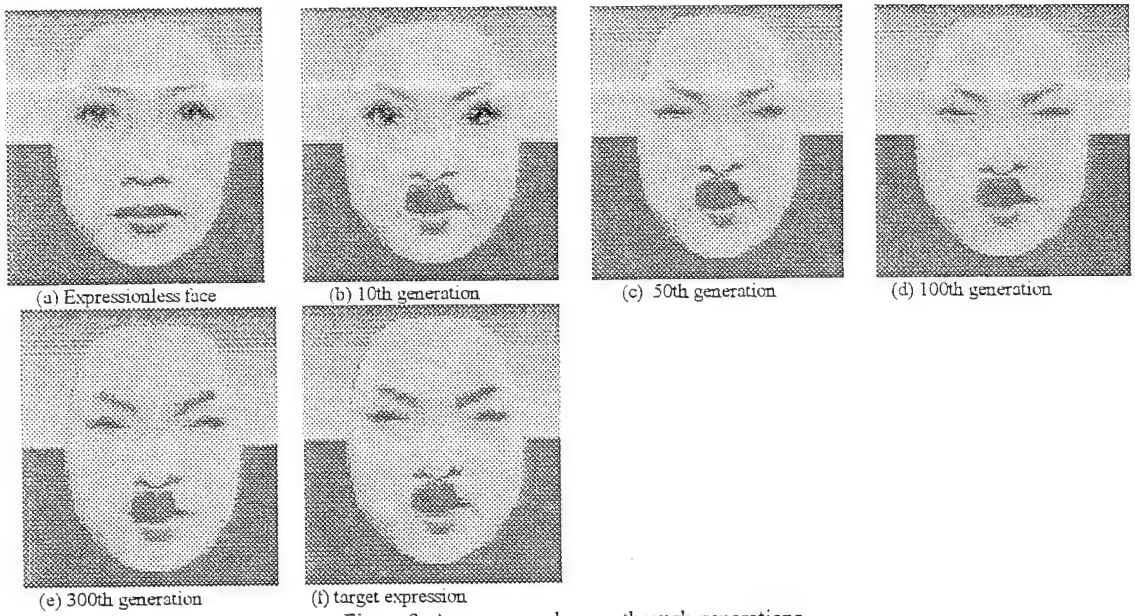


Figure 9. Appearance changes through generations

3. Conclusions

It is necessary to find out the combination of action units, the components of a facial expression, and their intensities for each expression to build natural looks. This paper presents a search method for the parameters of the composing elements of the facial expressions automatically. The facial expressions are the reflections of the human emotion and one's thought. Therefore, to find out the components of them means constructing a basis for creating a medium possessing the human feeling which connects man with a machine.

Especially in VR application, constructing a realistic facial expression is an indispensable part of building an artificial agent in a virtual world and this increases the realism of a medium which links a real world with a virtual one. It makes the environment of the man-machine interface more familiar to man. For instance, if the facial expressions are used as means of communication between the guide machine and the users, it can eliminate the user's feeling of rejection which often arises when we face the machine. A facial expression is an effective tool for a man to communicate with the other humans, can be used to represent an inner state of a system whose operation state is difficult to estimate and understand in a single glance.

In addition, in the view point of information transmission, it helps to save the amount of data to be transferred. When it is necessary to transfer facial expressions, if all their vertex data or the differences of the coordinates of the vertices between them are to be transferred, there will be a large amount of data and this may cause congestion in the communication channel even with advanced image compression methods like MPEG **II**. So if we can transfer only control parameters of the action units to change facial expressions with one basic expression, we can expect a large data compression. Moreover, our system can be modified to be used to analyze the psychological state of a man. If the data of a facial expression are given , we can extract the components of an expression and infer the emotional state of the expression.

With this purposes in mind, this research focuses on generating facial expressions at our will by implementing AUs which will describe the motion of the muscles of man more accurately.

References

[1] Paul Ekman and Wallace V.F. *Facial Action Coding System*. Consulting Psychologists Press Inc. (1978)

[2] Heedong Ko, Moon-Sang Kim, Hyun-Goo Park and Seung-Woo Kim. Face sculpturing robot with recognition capability. *CAD* vol26, NO. 11, 1994

[3] D. E. Goldberg, *Genetic Algorithms in search, optimization & Machine Learning*.Addison-Wesley (1989)

[4] D. Beasley, D. R. Bull, R. R. Martin. An Overview of Genetic Algorithms. *University Computing*, 1993, 15 (2) 58-69

[5] D. Beasley, D. R. Bull, R. R. Martin. Sequential Niche Technique for Multimodel Funciton Optimization. *Evolutionary Computaion* 1(2), (1993), pp101-125, MIT PRESS

[6] R. A Rutenbar. Simulated annealing algorithms : An Overview. *IEEE Circuits and Devices Magazine*, pages 19-26, January 1989

[7] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, Optimization by Simulated annealing, *Science*, vol.220, no.4598, pp.671-680,May 1983

[8] D.Whitley. A Genetic Algorithm Tutorial. Technical Report CS-93-103. Nov 10,1993

[9] J.H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 1975

[10] De Jong,K.A. An analysis of the behavior of a class of genetic adaptive systems. Dissertation abstracts International 36(10), 5140B. (University Microfilms No.76-9381)

COORDINATING VOCAL AND VISUAL PARAMETERS FOR 3D VIRTUAL AGENTS

Catherine Pelachaud
Dept. of Computer Science and Systems
University of Rome "La Sapienza"
pelachau@graphics.cis.upenn.edu

Scott Prevost*
Dept. of Computer and Information Science
University of Pennsylvania
prevost@linc.cis.upenn.edu

Abstract

This paper presents an implemented system for automatically producing prosodically appropriate speech and corresponding facial expressions for animated, three-dimensional agents that respond to simple database queries in a 3D virtual environment. Unlike previous text-to-facial animation approaches, the system described here produces synthesized speech and facial animations entirely from scratch, starting with semantic representations of the message to be conveyed, which are based in turn on a discourse model and a small database of facts about the modeled world.

1 Introduction

As research on the simulation of autonomous virtual human agents progresses, two major issues in human-machine interaction must be addressed. First, proper intonation is necessary for conveying the information structure of utterances with respect to the underlying discourse structure, expressing important distinctions of contrast and focus ([27], [24], [25]). Realistic facial expressions and lip movements help in providing relevant information about discourse structure, turn-taking protocols and speaker attitudes ([8], [9], [18]). Moreover, in a face-to-face conversation, facial displays play an important communicative role.

Simulating this communicative role for animation requires symbolic specification of the semantics and pragmatics of movements. Faces change expressions continuously, and many of these changes are synchronized with what is going on in concurrent conversation. Facial expressions are linked to the content of speech (scrunching one's nose when talking about something unpleasant) as well as affect (smiling when remembering a happy event). They can replace sequences of words (e.g. "the food was [wrinkle nose, stick out tongue]") as well as accompany them [9], and they can serve to help disambiguate what is being said when the acoustic signal is degraded. They do not occur randomly but rather are synchronized to one's own speech, or to the speech of others [6], [15]. It is therefore important that the specification of facial expressions takes many different levels of organization into account. We propose that integrating models for generating proper intonation and facial expressions will improve the intelligibility and naturalness of utterances produced by meaning-to-speech systems as well as by more elaborate systems involving virtual animated human agents (e.g. [3]).

The intonation generation model is based on Combinatory Categorial Grammar (CCG – cf. [27]), a formalism which easily integrates the notions of syntactic constituency, prosodic phrasing and information structure. Based on the CCG grammar, a simple discourse model and a small knowledge base represented in Prolog, the system produces spoken responses to database queries with appropriate intonation. Given the precise timings for phonemes and intonational phenomena in the speech wave, we produce precise specifications for generating the lip movements and facial expressions for a graphical model of a human head. Results from our current implementation demonstrate the system's ability to generate a variety of intonational possibilities and facial animations for a given sentence depending on the discourse context.

Previous work in the area of intonation generation includes studies by Terken ([29]), Houghton and Pearson ([13]), Isard and Pearson ([14]), Davis and Hirschberg (cf. [7], [12]), and Zacharski *et al.* ([31]). Benoit *et al.* ([1]),

*We would like to thank particularly Dr. Norman I. Badler and Dr. Mark Steedman for their very useful comments. We are grateful to AT&T Bell Laboratories for allowing us access to the TTS speech synthesizer, and to Mark Beutnagel, Julia Hirschberg, and Richard Sproat for patient advice on its use. The usual disclaimers apply. The research was supported in part by NSF grant nos. IRI90-18513, IRI90-16592, IRI91-17110 and CISE IIP-CDA-88-22719, DARPA grant no. N00014-90-J-1863, and ARO grant no. DAAL03-89-C0031.

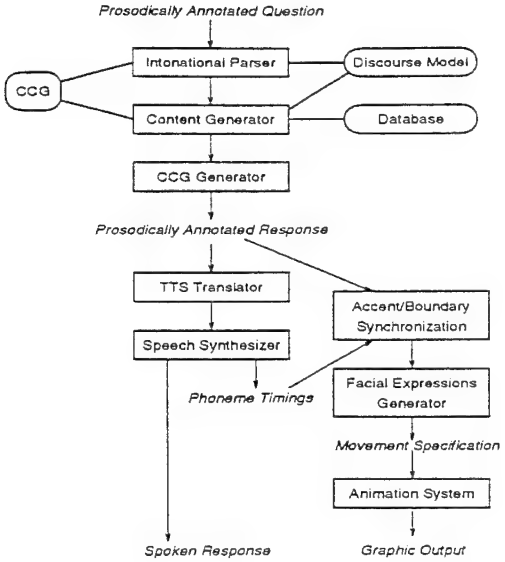


Figure 1: Architecture

Brooke ([2]), Cohen *et al.* ([4]), Hill *et al.* ([11]), Lewis *et al.* ([16]) and Terzopoulos *et al.* ([30]) have worked on synchronizing lip movements with speech, producing quite striking results. Takeuchi *et al.* ([28]) implemented a user-interface in which a 3D facial model responds to queries posed by a user. In this system, the generation of the facial expressions accompanying the answer depends on an analysis of the conversational situation and the selection of facial expressions from a database of facial displays.

The system described here expands the work of the aforementioned researchers by linking contextually appropriate intonation with the corresponding facial expressions, and generating the 3D facial animations automatically from semantic, information structural and discourse structural representations [21].

2 The Implementation

Using the CCG theory of prosody outlined in [27], [24] and [25], the implemented system undertakes the task of specifying contextually appropriate intonation and facial animation for spoken responses to database queries. The process, which is illustrated in figure 1, begins with a fully segmented and prosodically annotated representation of a spoken query, as shown in example (1), which involves a simple database of facts about stereo components. The notational system representing the intonation contour in example (1) is an adaptation of the widely used system developed by Pierrehumbert ([23]).[1] For simplicity, we show accented words in capital letters without regard for the different possible types of accents. A simple CCG parser determines the semantics of the question, dividing it into its *theme*, which identifies what the sentence is about, and its *rheme*, which identifies what is important or salient about the theme. We refer to this division of the utterance into theme and rheme as its *information structure*. Certain elements of the theme and rheme may be particularly salient because they are new to the discourse or serve to distinguish among entities or propositions that are already firmly established in the discourse. We say such items are in *focus*, and mark them with the * operator, as shown in examples (2).[2]

(1) I know which components produce MUDDY bass,
but WHICH components produce CLEAN bass?
L+H* LH% H* LL$

[1] The L+H* and H* markings represent different types of pitch accents in the fundamental frequency contour. The LH% and LL$ markings represent prosodic boundaries. For a brief explanation of the Pierrehumbert-style markings, see [26].

[2] A full explanation of the semantic and syntactic representation in (2) is beyond the scope of this paper. The interested reader should refer to [27] and [26].

(2) Proposition:
$s : \lambda x.component(x)\&produce(x, *clean(bass))$
Theme:
$s : \lambda x.component(x)\&produce(x, *clean(bass))/$
$(s : produce(x, *clean(bass))\backslash np : x)$
Rheme:
$s : produce(x, *clean(bass))\backslash np : x$

The content generation module has the task of determining the semantics and information structure of the response, marking focused items based on the contrastive stress algorithm described in [25]. For the question given in (1), the strategic generator produces the representation for the response shown in example (3), where the appropriate theme can be paraphrased as "what produces clean bass", the appropriate rheme as "amplifiers", and where the context includes alternative components and audio qualities.

(3) Proposition:
$s : produce(*amplifiers, *clean(bass))$
Theme:
$s : produce(x, *clean(bass))\backslash np : x$
Rheme:
$np : *amplifiers$

Using the output of the content generator, the CCG generation module (described in [24]) produces a string of words and Pierrehumbert-style markings representing the response, as shown in example (4).

(4) AMPLIFIERS produce CLEAN bass.
H* L L+H* LH%

The final aspect of speech generation involves translating such a string into a form usable by a suitable speech synthesizer. The current implementation uses the Bell Laboratories TTS system [17] as a post-processor to synthesize the speech wave and produce precise timing specifications for phonemes. The duration specifications are then automatically annotated with pitch accent peaks and intonational boundaries in preparation for processing by the facial expression rules (see also [3]).

Most facial animation systems use the Facial Action Coding System (**FACS**), developed by Ekman and Friesen [10], to annotate facial action. The system describes the visible muscular action based on anatomical studies, using basic elements called action units (**AU**), which refer to the contraction of one muscle or a group of related muscles. A facial expression is described as a set of **AU**s.

Certain facial expressions, which serve *informational structural* functions, accompany the flow of speech and are synchronized at the verbal level. Facial movements (such as raising the eyebrows or blinking while saying "AMPLIFIERS produce CLEAN bass") can appear during accented syllables or pauses. These function are based on the following determinants: conversational signals, punctuators and manipulators. *Conversational signals* correspond to movement occurring on accented or emphatic items to clarify or support what is being said. These can be eyebrow movements (the most commonly used facial expression), head nods, or blinks. *Punctuators* are movements which occur on pauses, reducing the ambiguity of the speech by grouping or separating sequences of words into discrete unit phrases [5]. Slow head movement, blinks, or a smile can accompany a pause. *Manipulators* correspond to biologically necessary functions like blinking to wet the eyes.

As we have seen, a facial expression can have a variety of different meanings (e.g. accentuating an element, punctuating a pause). We propose a high level programming language to describe them, amounting to a formal notation for the different clusterings of facial expressions. Indeed, rather than using a set of **AU**s to specify facial expressions in terms of intonational features in speech, it is more convenient to express them at a higher level, directly denoting their function. These operations are then mapped onto sequences of **AU**s so that we are able to model different facial "styles", in the sense that people differ in their way of emphasizing a word and in the number of facial displays they use. For example, Ekman [9] found that most people use raised eyebrows to accompany an accent while the actor Woody Allen uses eyebrow positions (inner and downward) which generally imply sadness.

Our algorithms incorporate synchrony ([6]), create coarticulation effects, emotional signals, and eye and head movements ([19], [20]). The facial animation system scans the input utterances and computes the associated movements

for the lips, the conversational signals and the punctuators. Conversational signals start and end with the accented word. For instance, on *amplifier*, the brow starts raising on 'a', remains raised until the end of the word, and ends raising on 'r'. On the other hand, the punctuator signals, such as smiling, coincide with pauses. Blinking is synchronized at the phoneme level, due to biological necessity, accentuation or pausing. On *amplifier*, for example, the eyes start closing on 'a', remain closed on 'm' and start opening on 'p'.

The computation of the lip shape is done in three passes. First, phonemes, which are characterized by their degree of deformability, are processed one segment at a time using the look-ahead model to search for the proximal deformable segments whose associated lip shapes influence the current segment. For example, in *amplifier* the 'l' receives the same lip shape as the following vowel 'i'—that is, the movement of the 'i' begins before the onset of its sound. Second, the spatial properties of muscle contractions are taken into account by adjusting the sequence of contracting muscles when antagonistic movements succeed one another (i.e. movements involving very different lip positions, such as pucker movements versus the extension of the lips). And finally, the temporal properties of muscle contractions are considered by determining whether a muscle has enough time to contract before (or relax after) the surrounding lip shape.

The tongue, although not highly visible, is an important element of distinction between phonemic elements, especially when these elements are not differentiated by their lip shapes. The tongue is composed of 2 parallel surfaces, each of them made of 10 triangles. A tongue shape is defined by varying the tongue parameters, including the length of the edges of the triangles and the angles between each of the edges. Modifying the length of the edges allows for the narrowing, flattening, stretching and/or compression of the tongue, while changing the value of the angles between edges allows the tongue to bend, curve and/or twist. This model is a simplification of [22].

3 Examples

In the examples shown below, the speaker manifests different behaviors depending on whether s/he is asking a question, making a statement, accenting a word or pausing. When asking a question, the speaker raises the eyebrows and looks up slightly to mark the end of the question. When replying, or when turning over the floor to the other person, the speaker turns the head toward the listener. To emphasize a particular word, s/he raises the eyebrows and/or blinks. During the brief pauses at the end of statements and within statements, the speaker blinks and smiles.

(5) I know which amplifier produces clean BASS,

but which amplifier produces clean TREBLE?
L+H* LH% H* LL$

The BRITISH amplifier produces clean TREBLE.
H* L L+H* LH$

(6) I know which British component produces MUDDY treble,

but which British component produces CLEAN treble?
L+H* LH% H* LL$

The British AMPLIFIER produces CLEAN treble.
H* L L+H* LH$

In utterance (5), the word *British* is accented and accompanied by a raised eyebrow, which indicates a conversational signal denoting contrast. In utterance (6), on the other hand, the word *amplifier* is accented and marked by the action of the eyebrows and a blink (see figure 2). The same argument differentiates the appearance of the movement on the word *treble* in (5) and the word *clean* in (6). Moreover, a punctuating blink marks the end of (6), starting on the pause after the word *treble*. In (5) a blink coincides with the accented word *treble* (as a conversational signal) and with the pause marking the end of the utterance (as a punctuator), resulting in two blinks emitted in succession at the end of the utterance. In both examples, the pause between the two intonational phrases '*the British amplifier*' and '*produces clean treble*', is accompanied by movement of the eyebrows.

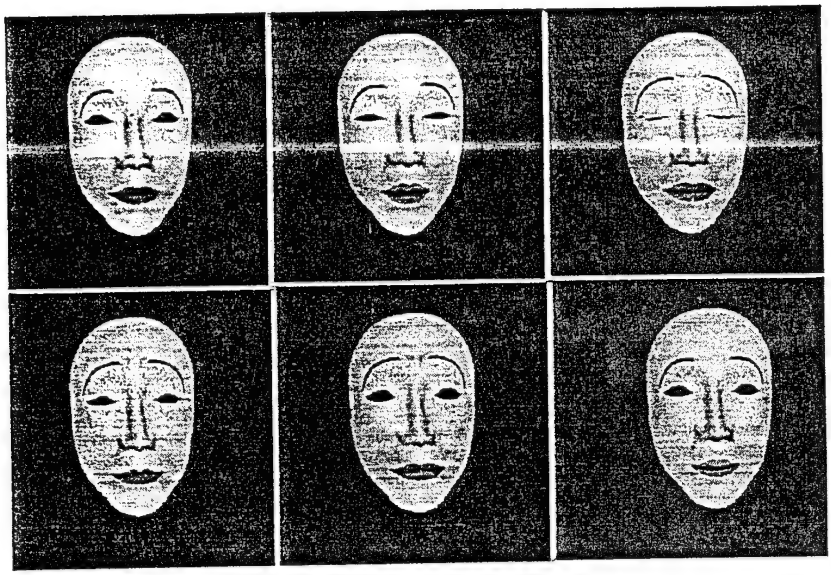


Figure 2: 'amplifier'

4 Conclusions

The system described above produces quite sharp and natural-sounding distinctions of intonation contour, as well as visually distinct facial animations, for minimal pairs of queries and responses generated automatically from a discourse model and a simple knowledge base. The examples in the previous section (and others presented at the workshop) illustrate the system's capabilities and provide a sound basis for exploring the role of intonation and facial expressions in a 3D virtual environment. Future areas of research include evaluating results and exploring the relevance of our current system to large scale animation systems involving autonomous virtual human agents (cf. [3]).

5 References

[1] C. Benoit. Why synthesize talking faces? In *Proceedings of the ESCA Workshop on Speech Synthesis*, pages 253–256, Autrans, 1990. ESCA.

[2] N.M. Brooke. Computer graphics synthesis of talking faces. In *Proceedings of the ESCA Workshop on Speech Synthesis*, Autrans, 1990. ESCA.

[3] J. Cassell, C. Pelachaud, N. Badler, M. Steedman, B. Achorn, T. Becket, B. Douville, S. Prevost, and M. Stone. Animated conversation: Rule based generation of facial expression, gesture and spoken intonation for multiple conversational agents. In *SIGGRAPH'94*, 1994.

[4] Michael M. Cohen and Dominic W. Massaro. Modeling coarticulation in synthetic visual speech. In D. Thalmann N. Magnenat-Thalmann, editor, *Computer Animation '93*. Springer-Verlag, 1993.

[5] G. Collier. Emotional expression. *Lawrence Erlbaum Associates*, 1985.

[6] W.S. Condon and W.D. Osgton. Speech and body motion synchrony of the speaker-hearer. In D.H. Horton and J.J. Jenkins, editors, *The perception of Language*, pages 150–184. Academic Press, 1971.

[7] J. Davis and J. Hirschberg. Assigning intonational features in synthesized spoken discourse. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, pages 187–193, Buffalo, 1988.

[8] S. Duncan. Some signals and rules for taking speaking turns in conversations. In Weitz, editor, *Nonverbal Communication*. Oxford University Press, 1974.

[9] P. Ekman. About brows: emotional and conversational signals. In M. von Cranach, K. Foppa, W. Lepenies, and D. Ploog, editors, *Human ethology: claims and limits of a new discipline: contributions to the Colloquium*, pages 169–248. Cambridge University Press, Cambridge, England; New-York, 1979.

[10] P. Ekman and W. Friesen. Facial action coding system. *Consulting Psychologists Press*, 1978.

[11] D.R. Hill, A. Pearce, and B. Wyvill. Animating speech: an automated approach using speech synthesised by rules. *The Visual Computer*, 3:277–289, 1988.
[12] J. Hirschberg. Accent and discourse context: Assigning pitch accent in synthetic speech. In *Proceedings of AAAI: 1990*, pages 952–957, 1990.
[13] G. Houghton and M. Pearson. The production of spoken dialogue. In M. Zock and G. Sabah, editors, *Advances in Natural Language Generation: An Interdisciplinary Perspective, Vol. 1*. Pinter Publishers, London, 1988.
[14] S. Isard and M. Pearson. A repertoire of British English intonation contours for synthetic speech. In *Proceeding of Speech '88, 7th FASE Symposium*, pages 1223-1240, Edinburgh, 1988.
[15] A. Kendon. Some relationships between body motion and Speech. In A.W. Siegman and B. Pope, editors, *Studies in Dyadic Communication*, pages 177-210, 1972.
[16] J.P. Lewis and F.I. Parke. Automated lip-synch and speech synthesis for character animation. *CHI + GI*, pages 143–147, 1987.
[17] M. Liberman and A. L. Buchsbaum. Structure and usage of current Bell Labs text to speech programs. Technical Memorandum TM 11225-850731-11, AT&T Bell Laboratories, 1985.
[18] D.W. Massaro. *Speech perception by ear and eye: a paradigm for psychological inquiry*. Cambridge University Press, 1989.
[19] C. Pelachaud, N.I. Badler, and M. Steedman. Linguistic issues in facial animation. In N. Magnenat-Thalmann and D. Thalmann, editors, *Computer Animation '91*, pages 15–30. Springer-Verlag, 1991.
[20] C. Pelachaud, M.L. Viaud, and H. Yahia. Rule-structured facial animation system. In *IJCAI 93*, 1993.
[21] C. Pelachaud and S. Prevost. Sight and sound: generating facial expressions and spoken intonation from context. In *Proceedings of the Second ESCA Workshop on Speech Synthesis*, New Paltz, NY, 1994. ESCA.
[22] C. Pelachaud, C.W.A.M van Overveld and C. Seah. Modeling and animating the human tongue during speech production. In *Computer Animation '94*, Geneva, May, 1994.
[23] J. Pierrehumbert. The phonology and phonetics of English intonation. PhD Dissertation, MIT (Dist. by Indiana University Linguistics Club, Bloomington, IN).
[24] S. Prevost and M. Steedman. Generating contextually appropriate intonation. In *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics*, pages 332–340, Utrecht, 1993.
[25] S. Prevost and M. Steedman. Using context to specify intonation in speech synthesis. In *Proceedings of the 3rd European Conference of Speech Communication and Technology (EUROSPEECH)*, pages 2103–2106, Berlin, 1993.
[26] S. Prevost and M. Steedman. Specifying intonation from context for speech synthesis. *Speech Communication*, 15(1-2), pages 139–153, 1994.
[27] M. Steedman. Structure and intonation. *Language*, pages 260–296, 1991.
[28] A. Takeuchi and K. Nagao. Communicative facial displays as a new conversational modality. In *ACM/IFIP INTERCHI '93*, Amsterdam, 1993.
[29] J. Terken. The distribution of accents in instructions as a function of discourse structure. *Language and Structure*, 27:269–289, 1984.
[30] D. Terzopoulos and K. Waters. Techniques for realistic facial modelling and animation. In N. Magnenat-Thalmann and D. Thalmann, editors, *Computer Animation '91*, pages 45–58. Springer-Verlag, 1991.
[31] R. Zacharski, A.I.C. Monaghan, D.R. Ladd, and J. Delin. BRIDGE: Basic research on intonation in dialogue generation. Technical report, HCRC: University of Edinburgh, 1993. Unpublished manuscript.

Virtual Urban Environment for the Simulation of an Automated Electrical Cars Platoon in the Praxitele Project

Bruno Arnaldi, Rémi Cozot and Stéphane Donikian

IRISA
Campus de Beaulieu
35042 Rennes Cedex, FRANCE
arnaldi@irisa.fr, cozot@irisa.fr, donikian@irisa.fr

Keywords: Simulation, Behavioral Animation, Multi-Agent System

1 Introduction

The Praxitele Project has in charge the design of a new kind of transportation in an urban environment, which consists in a fleet of electric public cars. These public cars are capable of autonomous motion on certain displacements between stations. The realization of such a project requires experiments of the behavior of autonomous vehicles in the urban environment. Because of the danger of this kind of experiments in a real site, it is necessary to design a virtual urban environment in which simulations can be done.

Existing driving simulators [21, 28, 13, 11, 17] differ in two points with the Praxitele Project requirements:

- They are all realized to integrate a real driver in the simulation loop, and all the simulation is based on this aspect (movement restitution, realistic driving interface, ...).
- The simulated environment is not urban.

The Praxitele project is presented in more details in the next section. The section 3 presents a simulation platform which complies with our needs for the testing of automatic motion control algorithms of vehicles evolving in a complex environment. The section 4 is devoted to the general presentation of the virtual urban environment, while the section 5 will focus on the description of an automatically driven dynamic car. A first prototype of Real-Time Driving Simulation is shown in the section 6, and then in the last section we will discuss about the evolution of this work.

2 Aims and goals of the Praxitele Project

The Praxitele project is done by two large government research institutes, one in transportation technologies (INRETS), the other in computer science and automation (INRIA), in cooperation with large industrial companies (RENAULT, EDF, CGEA). This project designs a novel transportation system based on fleet of small electric public cars under supervision from a central computer [24]. These public cars are driven by their users but their operation can be automated in specific instances. The system proposed here should bring a solution to the congestion and pollution in most cities through the entire world.

The concept of a public transport system based on a fleet of small electric vehicles has already been the subject of experiments several times but with poor results. The failure of these experiments can be traced to one main factor : poor avaibility of the vehicles when a customer needs one. To solve this main problem, Praxitele project develops and implements automated cooperative driving of a platoon of

vehicles, only the first car is driven by a human operator [23]. This function is essential to move easily the empty vehicles from one location to another.

Before first experiments of real automated cars in real cities, we have to design and implement a simulation platform. This platform permits to simulate a platoon of vehicules evolving in a virtual urban environment and so to test control algorithms of the automated cars.

3 A Simulation Platform

Motion control models are the heart of any simulation system that determines the friendliness of the user interface, the class of motions and deformations produced, and the application fields. Motion control models can be classified into three general families : descriptive, generative and behavioral models [14]. Descriptive models are used to reproduce an effect without any knowledge about its cause, thus a subset of instantaneous states are expressed either absolutely or relatively over time, and by interpolation a spatio-temporal trajectory is obtained in the system description space. Unlike preceding models, generative models are interested by a causal description of objects movement (describe the cause which produces the effects) for instance their mechanics. The goal of behavioral models is to simulate organisms (plants) [8, 25] and living beings (animals and persons) [2, 3], their action and their response to stimulation. One goal of behavioral models is to provide the user with a higher level control of movement. With the intention of making these three kinds of models work together, we are interested by their integration into a same simulation platform. This integration of the three kinds of models in the same platform permits to offer to each dynamic entity a more realistic and richer environment, and thereby will increase possible interactions between an *agent/actor* and its environment.

The simulation platform is composed of a set of *agents/actors* whose synchronization and communication are managed by a real-time kernel (cf figure 1). The main part of this kernel is the general controller. Communication between the agents is both synchronous and asynchronous. The synchronous part is data-flow based where each agent has its own frequency and is managed by the general controller. So, the data-flow communication channels include all the mechanisms to adapt to the local frequency of the sender and receiver agents (over-sampling, sub-sampling, interpolation, extrapolation, etc...). The asynchronous part is based on event based communication between agents and the general controller.

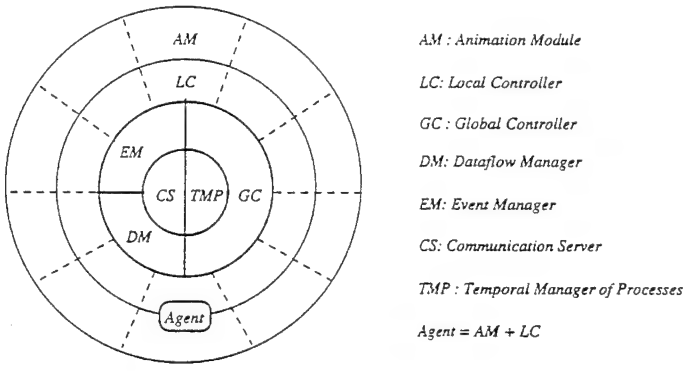


Figure 1: layers of the logical architecture of the platform

Time is the most important element of our simulation platform either during the specification or during the execution phase. The figure 2 show a structural view of two communicating agents in the platform. At each component is associated a specific task:

Global controller: it is the centralised controller of an animation/simulation. It performs the scheduling of the agents execution in order to respect the real time constraint. It is responsible of the initialization and dynamic configuration of the set of agents through communications with the agent using events. The dynamic configuration depends of events generated by the agents or of the analysis of an external script describing the animation/simulation.

Local controller: this controller has to manage the communications by events to the global controller and by data-flow to the other agents. The temporal control of the animation module is also performed by this controller according to the global controller directives.

Animation module: this module is the effective computation module performing the animation task as user interaction, physically based models calculation, trajectories application, image synthesis and so on. Each animation module has a local frequency according to its functionality.

Communication channel: the communication channel connects two agents with potentially different local frequencies. It has to adapt the data-flow communication according to these known frequencies by temporally stamping messages between agents.

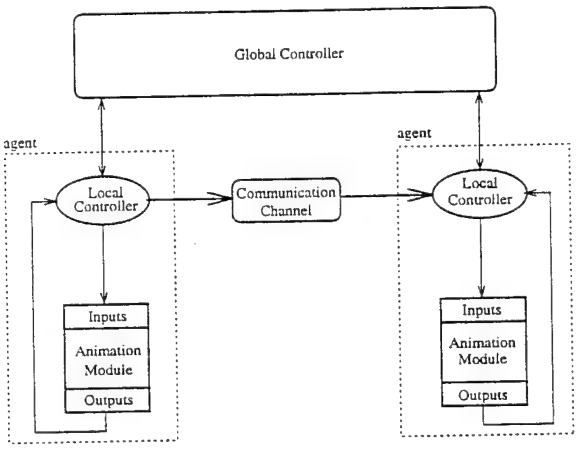


Figure 2: structural view of the platform

4 Simulation of an Urban Environment

An urban environment is composed of many dynamic entities evolving in a static scene. These dynamic entities have to be both autonomous and controllable and also realistic. It is necessary to combine the three motion control models to describe dynamic entities of the environment. For example, to describe traffic lights it is not necessary to use a generative model when a descriptive model (finite state automata) is sufficient. On the other hand, for a realistic car driving, we need both generative and behavioral models (the first one to simulate the dynamic of the vehicle and the second one to simulate the driver).

4.1 The static scene

As we want to control entities evolving, we need to link dynamic entities with the static scene in which they are moving. This link requires a semantic knowledge on the scene. If we want to simulate as completly as possible the life of a city, we need a lot of semantic informations. For the car driving simulation example, we are particularly interested by the life in the streets and less by what is happening inside buildings (cf 3). Informations required for the simulation are:

1. Geometric:
   - the geometry of the town,
   - roadsigns,
2. Topologic:

- the road network (network of trajectories),
- a visibility grid,

3. Semantic:
   - road informations: roadsigns, color of traffic lights, qualitative aspect of the road, ...
   - city informations: name of streets, quarters, particular buildings, squares, ...

To describe such a scene, an urban modeling system is currently in development. This system is an extension of the Scriptography (Declarative Design System) [10], in which geometric, topologic and semantic informations are mixed.

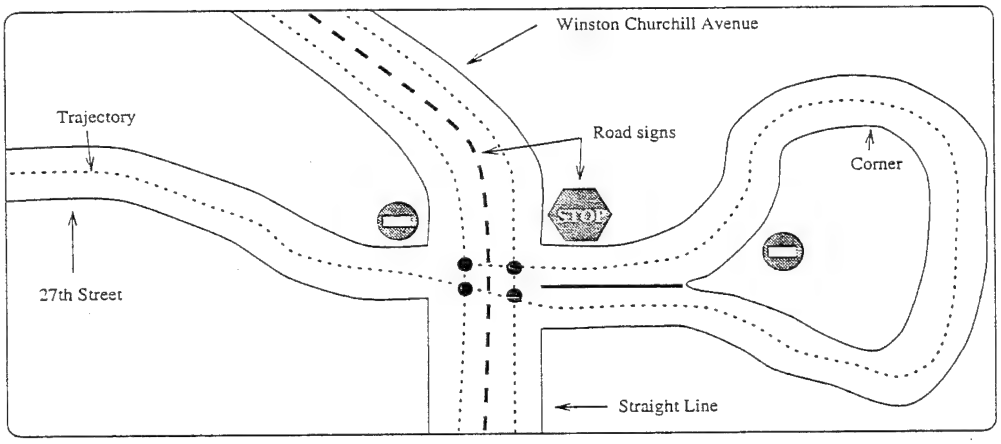


Figure 3: Geometric, Topologic and Semantic Informations

4.2 Dynamic entities

To take into account natural phenomena, the first work is to choose a physical model to represent the object. From a high level description of articulated rigid body systems, a simulation blackbox is generated whose inputs are torques and outputs are position and orientation parameters.

We have now to determine how to control this physical model that is to say, depending on the actual state of the entity what kind of torque must we apply to it to obtain the desired motion ? In the case of an automatic motion control, this question can be decomposed in two parts:

1. how control the physical model?
2. what is the desired motion?

The answer to the first question consists in using motion control algorithms [19] well known in the automatic and robotic communities, which can permit to build a library of elementary actions. The behavioral model try to answer to the second question by defining actions and reactions of an entity [31, 9, 16]. The behavioral model is based on two kinds of relationships between the object and its environment: perception and action (cf figure 4).

Different approaches have been studied for the decision part: Sensor-Effector [32, 34, 30], Behavior Rule [6, 15, 22, 26, 29, 33, 31], Predefined Environment [7, 27] and State Machine [4]. The general statement about actual systems using these different approachs is that they are *ad hoc* models designed to be applied in some particular cases, in which the simulated environment is very simple. Possible interactions between an object and its environment are very simple, and sensors and actuators are reduced to minimal capabilities which, most of the time, permit only to avoid obstacles in a 2D or 3D world. Another conclusion is that none of these models are able to take into account the explicit management

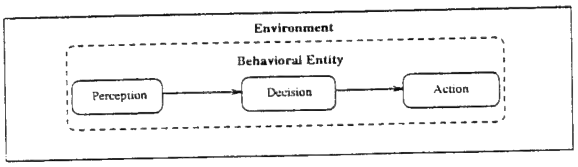


Figure 4: A behavioral agent immersed in its environment

of time, either during the specification phase (memorization, prediction, action duration, etc.) or during the execution phase (synchronization of objects with different internal times).

We have chosen to define a multi-agent system [12, 18] in order to implement a cooperation between the different behavioral approaches in one decisional model. So, this decisional model is decomposed in a set of specialized agents who use themselves some experts before proposing their diagnostic to the decisional agent (supervisor). The work of this supervisor is to integrate all the local decisions according to the desired behavior of the system. The supervisor is principally constituted of hierarchical parallel automata, whose transitions depend on sensor data and on event generated by lower level agents. Therefore, the supervisor decides to activate some of the specialized agents and this decision depends on its own state and on sensor data (cf figure 5). The use of hierarchical parallel automata allows us to take into account concurrency and abstraction in the description of the behavior, like Kearney et al. with their hierarchical concurrent control state machines [16]. Because of the integration of our behavioral model in a simulation platform, we have also the ability to deal with a real time during the specification and the execution phase.

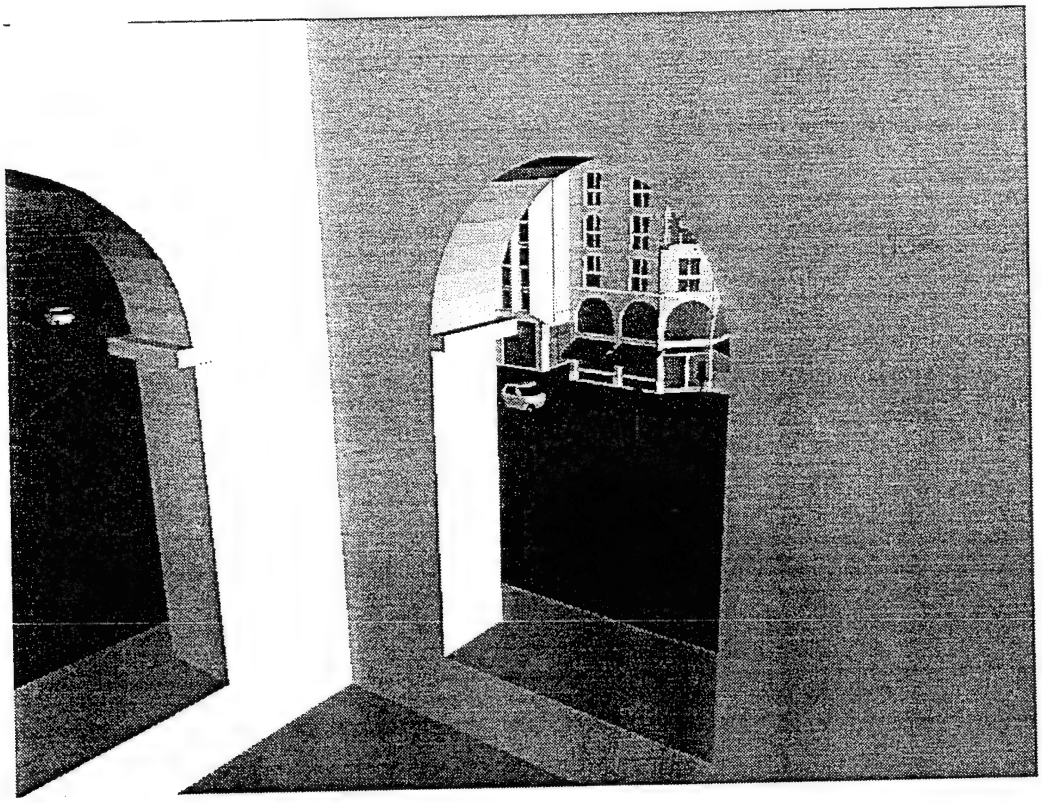


Figure 5: One point of view: one car is stopping at the crossroad because the traffic light is red, another one is now out of the corner.

5 An Automatically Driven Dynamic Car

A vehicle is an articulated rigid object structure if we do not consider the deformations of the tires and the component flexibility [1]. The vehicle is defined by a generative model which is parametrized by a state vector and two torques (motor and guidance). In the case of an automatic control of the vehicle, we have to describe the behavior of a virtual driver depending on how is its perception of its environment. A feedback state control algorithm [20] determines what torques are applied to the vehicle from actions decided by the virtual driver.

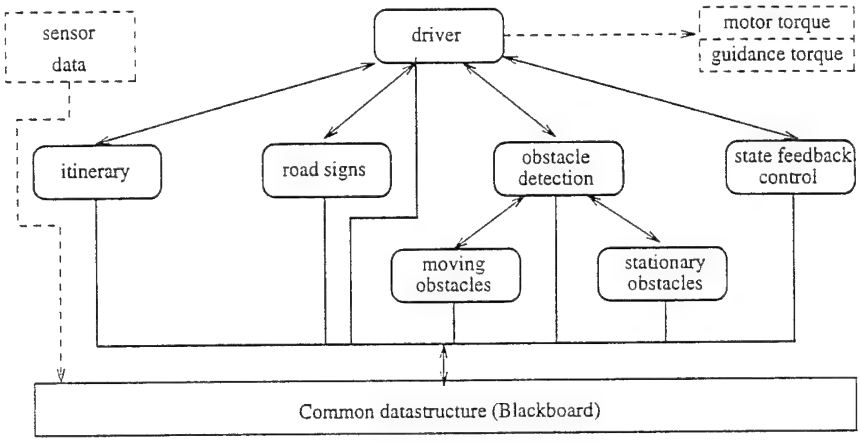


Figure 6: Hierarchical structure of the driver decisional model.

The decisional module has to make the choice of what kind of action to perform, depending on its actual state and on its own perception of its environment (cf figure 6). This operation is decomposed in six stages:

1. The supervisor (the driver module) reads received messages and makes an analysis of data received from sensor(s).
2. The supervisor activates specialized agents.
3. Execution of specialized agents (itinerary, road signs, obstacle detection and state feedback control) which can also activate, if necessary, more specialized agents (here the obstacle detection module can activate both the moving obstacles module and the stationary obstacles module).
4. The supervisor analyses diagnosis of specialized agents.
5. The supervisor decides to act.
6. *Actions.*

Stages two, three and four are corresponding to the calculation of the new state of the world from the point of view of the agent. To deal with complex and concurrent behaviors, stage five is performed by hierarchical parallel automata (cf figure 7). Transitions on automata are functional expressions depending on the actual state of the world. Each state of each automaton is either an automaton itself or an elementary state. Each elementary state has in charge to propose an action to execute.

The *road signs* module is, at the moment, in charge of determining the value of three parameters:

- speed limitation (real value),
- overtaking (YES | NO),
- crossroads priority (right of way | priority to the right way | stop | traffic lights),

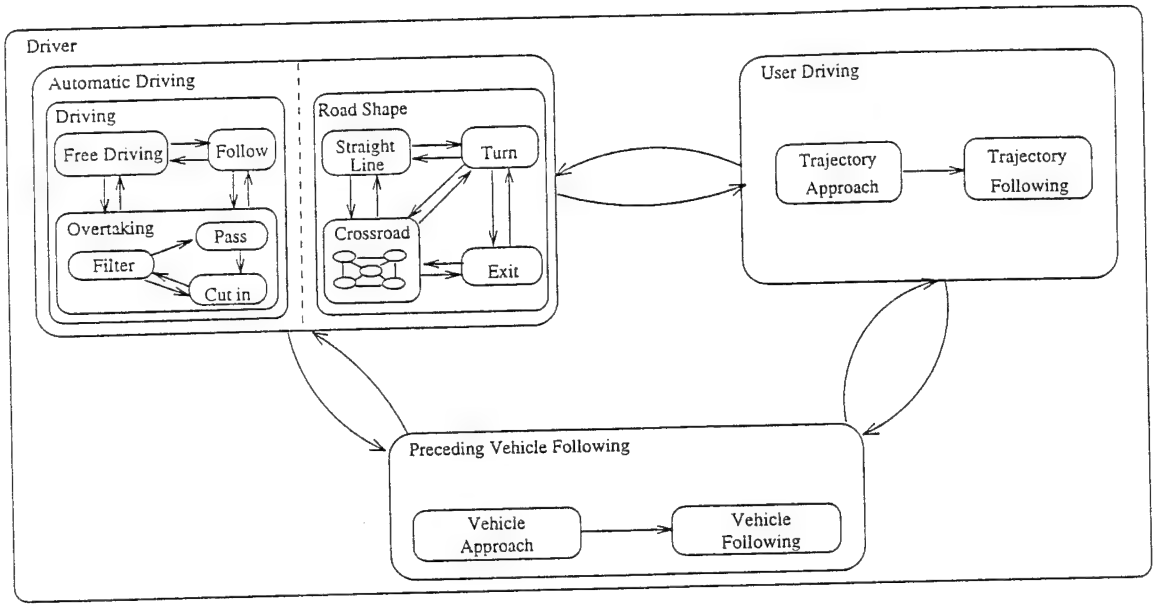


Figure 7: Hierarchical Parallel Automata of the Driver.

The *itinerary* module is in charge of determining the new direction at each crossroads, so this module is only activated when the vehicle is near one of them. The *obstacle detection* module has to determine if there is some possible intersections between the vehicle desired trajectory and predicted trajectories of other vehicles, and if so to propose a new trajectory. Actions managed by the *state feedback control* module are:

for the guidance torque:

- *follow_trajectory(actual position, desired trajectory, circle radius)*,

and for the motor torque:

- *accelerate(desired speed)*,
- *brake()*,
- *stop(distance)*,
- *cover(distance,delay)*,
- *follow(preceding vehicle actual speed, distance)*.

In order to simplify calculation, the human vision is not completely simulated but is replaced by a global knowledge on the scene geometry and on the location of objects, then by using visual sensors we obtain qualitative informations about objects in the vision cone.

6 A first Prototype of Real-Time Driving Simulation

A first implementation of this system has been realized as part of the car driving example. It is a modular system whose synchronization and control are specified in the synchronous real-time language SIGNAL [5]. Data communication between agents is realized by using the notion of Blackboard (common data structure).

We describe now an example of a little town composed of some different kinds of buildings (appartment buildings, separated houses, church), a eight shaped road, a crossroads with traffic lights and seven

ground vehicles (cf figure 8). There is three kinds of vehicles: one of them is driven by a user (by using a mouse) and corresponds to the first vehicle of the platoon; three of them represent other vehicles of the platoon and the three last one describe the dynamic environment.

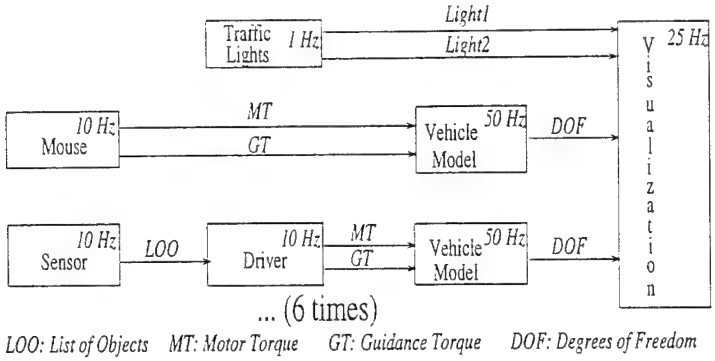


Figure 8: Driving example composition.

Internal frequencies of these modules are relatively different: the vehicle module has an internal frequency that can not be less than 50 Hz because of numerical convergence while sensors have an internal frequency of 10 Hz and the visualization frequency must be ideally of 25 Hz. The traffic lights module does not require a higher frequency than 1 Hz. It is necessary to synchronize the execution of all these modules to offer to real and virtual drivers a *realistic world*.

The figure 8 show synchronous dataflow communications between modules of the example. There is also some asynchronous event based communications between the mouse module and the others:

Mouse ⇒ **Traffic Lights:** A *reinitialization()* message can be sent at anytime, according to the user decision.

Mouse ⇒ **Visualization:** Different kinds of message are sent to the visualization module to control the point of view. For example, a message *set_view(front | behind | right | left | upon | driver)* changes the viewpoint whereas a message *set_vehicle(1 | 2 | ... | 7)* changes the view reference point.

Mouse ⇒ **Global Controller:** A message *termination()* is sent to the global controller when the user decides to terminate the simulation.

7 Futur Works

In the actual version of our system, the interaction of more than one user with the simulated environment is not possible. We are actually working on the generalization of this model to deal with communicating processes. Our goal is to permit the platform execution whatever the hardware configuration may be, and to offer a high quality multiuser interface. In addition, a general reflection is done to characterize a language which offer the ability to specify both agents and their different kinds of relations.

8 Conclusion

We have presented in this paper a platform which permit to simulate a line of autonomous vehicles in an urban environment. The integration of descriptive, generative and behaviorals models in the same simulation platform offer to each dynamic entity a more realistic and richer environment, and thereby increase possible interactions between an agent and its environment. This work is directly applied as part of the PRAXITELE project. Our work will permit to first simulate the line of vehicles evolving in a virtual environment with the intention of giving experimenters the ability to test their control algorithms whose inputs are informations from virtual sensors.

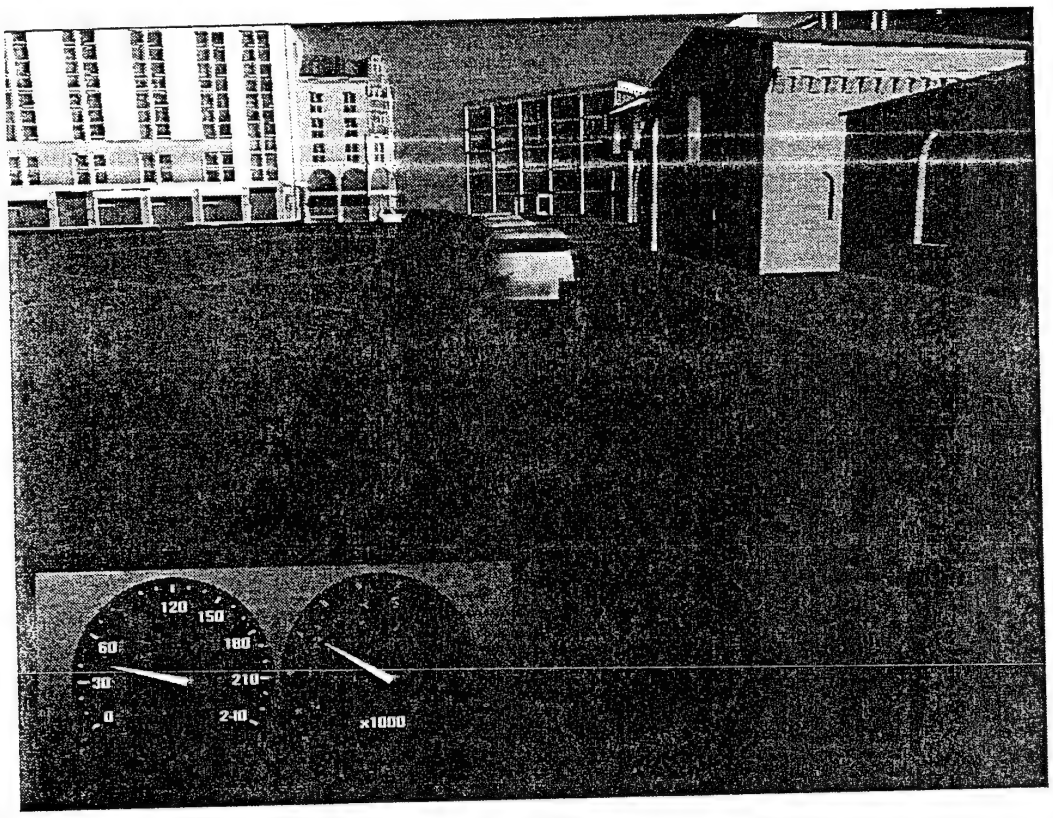


Figure 9: One example of obtained simulation: the traffic light being red on his way, the user decides to stop the first car of the platoon, then the three others decelerate also, while an automatically driven car is passing through the crossroad.

References

[1] B. Arnaldi and G. Dumont. Vehicle simulation versus vehicle animation. In Proceedings of *Third Eurographics Workshop on Animation and Simulation*, Cambridge, Sep. 1992.

[2] N. I. Badler, C. B. Phillips, and B. L. Webber. *Simulating Humans : Computer Graphics Animation and Control.* Oxford University Press, 1993.

[3] N. I. Badler, B. L. Webber, J. Kalita, and J. Esakov, editors. *Making them move: mechanics, control, and animation of articulated figures.* Morgan Kaufmann, 1991.

[4] M. Booth, J. Cremer, and J. Kearney. Scenario control for real-time driving simulation. In Proceedings of *Fourth Eurographics Workshop on Animation and Simulation*, pages 103–119, Politechnical University of Catalonia, Sep. 1993.

[5] P. Bournai, B. Chéron, T. Gautier, B. Houssais, and P. Le Guernic. *Signal Manual.* Technical Report 745, IRISA, July 1993.

[6] B. Coderre. Modeling behavior in petworld. In Proceedings of *artificial life*, pages 407–420, Addison-Wesley, 1988.

[7] M. F. Cohen. Interactive spacetime control for animation. In Proceedings of *Computer Graphics (SIGGRAPH '92 Proceedings)*, E. E. Catmull, editor, pages 293–302, July 1992.

[8] P. de Reffye, C. Edelin, J. Francon, M. Jaeger, and C. Puech. Plant models faithful to botanical structure and development. In Proceedings of *Computer Graphics (SIGGRAPH '88 Proceedings)*, J. Dill, editor, pages 151–158, Aug. 1988.

[9] S. Donikian and B. Arnaldi. Complexity and concurrency for behavioral animation and simulation. In Proceedings of *Fifth Eurographics Workshop on Animation and Simulation*, G. Hégron and O. Fahlander, editors, Oslo, Norvège, Sep. 1994.

[10] S. Donikian and G. Hégron. A declarative design method for 3d scene sketch modeling. In Proceedings of *EUROGRAPHICS'93 Conference Proceedings*, Barcelona, Spain, Sep. 1993.

[11] P. Gauriat and D. Lechner. Les simulateurs pour la recherche dans le domaine de l'automobile. *Recherche Transports Sécurité*, (30):43–51, 1991.

[12] J. P. Haton, N. Bouzid, F. Charpillet, M. C. Haton, B. Lâasri, H. Lâasri, P. Marquis, T. Mondot, and A. Napoli. *Le raisonnement en intelligence artificielle. Modèles techniques et architectures pour les systèmes à bases de connaissances.* InterEditions, 1991.

[13] E. Haug. *Feasability Study And Conceptual Design of a National Advanced Driving Simulator.* Technical Report, University of Iowa, 1990.

[14] G. Hégron and B. Arnaldi. *Computer Animation : Motion and Deformation Control.* Eurographics'92 Tutorial Notes, Eurographics Technical Report Series, Cambridge (Grande-Bretagne), Sep. 1992.

[15] D. Kalra and A. Barr. Modeling with time and events in computer animation. In Proceedings of *Eurographics*, A. Kilgour and L. Kjelldahl, editors, pages 45–58, Blackwell, Cambridge, United Kingdom, Sep. 1992.

[16] J. Kearney, J. Cremer, and S. Hansen. Motion control through communicating, hierarchical state machines. In Proceedings of *Fifth Eurographics Workshop on Animation and Simulation*, G. Hegron and O. Fahlander, editors, Oslo, Norway, Sep. 1994.

[17] A. Kemeny and J. Piroird. A simulator for cooperative driving. In Proceedings of *Drive Conference*, Bruxelles, 1991.

[18] I. Lapierre, C. Laurgeau, J. Drappier, and M. Vezzoli. Un automate de compréhension de l'environnement routier pour l'assistance à la conduite automobile. In Proceedings of *L'interface des mondes réels & virtuels*, pages 533–548, Montpellier, March 1992.

[19] C. Lecerf. *Contrôle du mouvement de systèmes mécaniques en animation.* PhD thesis, Université de Rennes 1, Sep. 1994.

[20] C. Lecerf, B. Arnaldi, and G. Hégron. Mechanical systems motion control for computer animation. In Proceedings of *First Bilkent Computer Graphics Conference on Advanced Techniques in Animation, Rendering and Visualization, ATARV-93, July 12-14, 1993*, B. Ozguc and V. Akman, editors, pages 207–222, Bilkent University, Ankara, Turkey, jul 1993.

[21] G. Malaterre and D. Lechner. *Expérimentation de manœuvres d'urgence sur simulateur de conduite, comportement des conducteurs.* Technical Report 104, INRETS, Nov. 1989.

[22] R. Mouli, S. Duthen, and R. Caubet. Un modèle de simulation comportementale orienté objet pour l'animation intelligente. In Proceedings of *Interface des mondes réels & virtuels*, pages 111–121, Montpellier, France, March 1992.

[23] M. Parent and P. Daviet. Automatic driving for small public urban vehicles. In Proceedings of *Intelligent Vehicle Symposium*, Tokyo, Japon, July 1993.

[24] M. Parent and P. Texier. A public transport system based on light electric cars. In Proceedings of *Fourth International Conference on Automated People Movers*, Irving, Texas, U.S.A., March 1993.

[25] P. Prusinkiewicz, M. S. Hammel, and E. Mjolsness. Animation of plant development. In Proceedings of *Computer Graphics (SIGGRAPH '93 Proceedings)*, J. T. Kajiya, editor, pages 351–360, Aug. 1993.

[26] C. W. Reynolds. Flocks, herds, and schools: a distributed behavioral model. In Proceedings of *Computer Graphics (SIGGRAPH '87 Proceedings)*, M. C. Stone, editor, pages 25–34, July 1987.

[27] G. Ridsdale and T. Calvert. Animating microworlds from scripts and relational constraints. In Proceedings of *Computer Animation '90 (Second workshop on Computer Animation)*, N. Magnenat-Thalmann and D. Thalmann, editors, pages 107–118, Springer-Verlag, Apr. 1990.

[28] T. Suetomi, A. Horyguchi, Y. Okamoto, and S. Hata. *The Driving Simulator with Large Amplitude Motion System.* Technical Report 910113, SAE, 1991.

[29] H. Sun and M. Green. The use of relations for motion control in an environment with multiple moving objects. In Proceedings of *Graphics Interface*, pages 209–218, Toronto, Ontario, May 1993.

[30] M. Travers. Animal construction kits. In Proceedings of *artificial life*, pages 421–442, Addison-Wesley, 1988.

[31] X. Tu and D. Terzopoulos. Artificial fishes: physics, locomotion, perception, behavior. In Proceedings of *Computer Graphics (SIGGRAPH'94 Proceedings)*, pages 43–50, Orlando, Florida, July 1994.

[32] M. van de Panne and E. Fiume. Sensor-actuator networks. In Proceedings of *Computer Graphics (SIGGRAPH '93 Proceedings)*, J. T. Kajiya, editor, pages 335–342, Aug. 1993.

[33] T. Widyanto, A. Marriott, and M. West. Applying a visual perception model to a behavioural animation system. In Proceedings of *Eurographics Workshop on Animation and Simulation*, pages 89–98, Vienna, Austria, Sep. 1991.

[34] J. Wilhelms and R. Skinner. A "notion" for interactive behavioral animation control. *IEEE Computer Graphics and Applications*, 10(3):14–22, May 1990.

AN INTERACTIVE VIRTUAL WORLD EXPERIENCE - THE CYBERSPACE ROADSHOW

Peter Astheimer and Wolfgang Felger

Fraunhofer-Institut für Graphische Datenverarbeitung (IGD)
Demonstration Center for Virtual Reality
Wilhelminenstr. 7, D-64283 Darmstadt, Germany
Phone: ++49 6151 155 121 & 122, Fax: ++49 6151 155 399
Email: {astheime, felger}@igd.fhg.de

ABSTRACT

The goal of this paper is to describe a virtual reality application which requires a mobile installation, and the experience gathered during the preparation, realization, and execution of this project.

Recently, high-end virtual reality (VR) has received much attention in the marketing field. Professional advertisment agencies are trying to exploit the fascination in this technology raised within the public. IGD was approached by such an agency and requested to prepare a high-quality VR application, serving as a marketing event for the Schweizerische Bankgesellschaft / Union de Banques Suisses (SBG/UBS), the largest Swiss bank. SBG wanted to attract the Swiss youth in order to promote its Junior Bank Card.

This event was held in twelve major Swiss cities during four weeks in May/June, 1994, with daily presentations. The time schedule and the marketing concept required a mobile installation. As "Cyberspace Roadshow" the event was advertised in newspapers and radio spots. This was worldwide the first mobile, immersive, high-quality VR installation. The complete VR infrastructure was installed in a large truck. The presentations were performed inside the truck as well. The hardware infrastructure was comprised of an image generator (SGI Crimson/RealityEngine), VR peripherals (head-mounted display, data glove, body tracking systems), a sound generator (SGI Indigo), and a large , stereoscopic rear projection screen using polarizing light. Within such an immersive environment, approximately 40 persons can be involved. Wearing 3D glasses the audience can experience stereoscopic viewing and follow, passively, what one active person is controlling by utilizing the VR devices.

For this event IGD created several entertaining virtual worlds using its proprietary VR system. The main idea was that one active player is flying through a tunnel, approaching a switch room with three alternatives: riding through a jungle, playing musical instruments or exploring a space labyrinth. Each player had a time limit of approximately 5 minutes.

Keywords: virtual reality, roadshow, case study

INTRODUCTION

Almost continuously during the last couple of years since the foundation of IGD's virtual reality demonstration center a variety of VR presentations have been realized. The presentations cover demonstrations at scientific conferences, in-house consulting, museum exhibitions as well as fairs and commercial exhibitions. The main goal of all performed contract-based VR presentations was to attract an audience by featuring VR technology. This shows the potential of VR as a marketing instrument in business management (Felger and Waehlert 1995).

In late 1993 the advertising agency Bosch & Butz approached IGD with the concept idea of a special marketing event for the Schweizerische Bankgesellschaft (SBG), the largest Swiss banking institute. In order to address its young clients, the Swiss youth, the SBG has currently a marketing concept which concentrates on the realization of a few large and unique events per year rather than numerous small ones. The motto for the year 1994 was "the year of adventures" and was to provide exciting and thrilling experiences. The previous event was the European premiere of the Disney movie "Aladdin" in an outdoor cinema-setup on top of a mountain in the Swiss Alps with a screen carved into a glacier. The basic idea for the next event was to exploit the fascination of VR technology for marketing and promoting the SBG. In several Swiss cities people were to be able to experience VR for themselves. The realization of a roadshow was the ultimate goal. Although the initial enthusiasm for VR has diminished, media

and the general public are still excited, open and eager for a new experience (Cohen 1994).

This paper represents a case study and is organized according to the following structure: First, preparation activities und basic thoughts are presented. Afterwards, realization aspects are discussed, showing the problems of the project. Later, experiences in the execution phase of the project gained during the public presentations are summarized. The paper closes with an outlook on future work and the conclusions.

PREPARATION PHASE

The project started in late February, 1994, and should have been on the road around mid May, 1994. This gave us less than three months for the project preparation and realization.

After initial negotiations about costs and considerations about the feasibility of the planned event we decided to take the chance to explore new terrain. Our partners at the bank and the advertising agency neither posed any restrictions on us nor enforced or suggested specific solutions (only the glove was a must, because it was the central component of the advertisement campaign, symbolizing VR). We "only" had to realize something exciting and thrilling. This left us with complete freedom for our work, but also caused some troubles and a lot of headaches.

We started to tackle the project with a team of computer graphics and design people (supported by students, of course). Although we only had few weeks left for designing and modeling the virtual world and specifying and integrating new features in our VR system, we were confident about the realization of the event. Soon we decided on the kind of the overall system interaction: exploring an adventure world. Shoot and quick reaction scenarios were abandoned because of ethical and technological (body tracking latency) reasons. Think and learn contents are not explicitly thrilling for everyone.

We had to find solutions for two basic challenge categories:

1. Logistic and technology challenges: For the purpose of this marketing event a mobile installation had to be planned which could tour some ten major Swiss cities during a month's time. Adequate locations had to be found and prepared (e.g., administration clearance, power supply, local support people, catering for guests). The installation had to be robust enough to run a continuous six hour show per day and endure the intermediate transport. Complicating the organization, three showmasters had to be trained because there are three different languages regions in Switzerland (French, German, and Italian). Furthermore, customs and insurance issues had to be solved.
2. System challenges: The experience had to be tailored for use by anyone with any education and any experience around the age of 17 (i.e., SBG's Junior Bank Card holders). The major demands for the virtual world and the system were:
   - Continuous motion in order to create interesting worlds. No pure walkthrough or flythrough.
   - Provide an atmosphere others than the pure geometric architecture of a world.
   - Dynamic worlds rather than static worlds.
   - Easy, intuitive interaction which doesn't requires training or longer introduction (self-explaining content).
   - Exciting and thrilling content (for users of this age/generation).
   - Participation of a larger audience (more than 20 people).
   - Quick turnaround times for players.

All these aspects had been taken into consideration in order to realize an attractive, mobile, immersive, high-quality VR installation. For further reading (Blinn 1994, Dodsworth 1994, Giles et al. 1994, and Helman 1994) are recommended.

REALIZATION PHASE

This chapter covers the realization aspects for the roadshow, dealing with hardware components, interaction issues, and the overall story of the presentation. The VR system used is IGD's Virtual Design (Astheimer et al. 1993).

Hardware Components

Fulfilling the requirements of touring accross the country and minimizing the time without presentations lead to the idea for a VR installation inside a large truck (see Fig. 1). This enables times of approximately one hour to start-up or shut-down the complete infrastructure at each show location. It was intented to run the shows in public, open-air places and in large halls. As a larger audience should be able to watch the active players and their behavior within the virtual worlds the truck was open at one side during show times. To avoid the brightness of sun light, the open side could be covered by a tent-like curtain.

The basic VR equipment for a player, including a powerful graphics workstation and

peripherals (head-mounted display, navigation device, tracking system), is very expensive. For this reason we decided to install one high-quality VR station for the active player and a stereoscopic large-screen rear projection, utilizing polarizing light, for passive observation by the audience (see Fig. 2). Figure 3 shows the VR infrastructure of the installation. It is comprised of an image generator (SGI Crimson/RealityEngine1 with Multi-Channel Option), VR peripherals (Virtual Research FlightHelmet, VPL DataGlove, Polhemus Fastrak), a sound generator (SGI Indigo), and stereoscopic rear projection (NEC/TAN). Within such an immersive environment, approximately 40 persons can be involved. Wearing 3D glasses the audience can experience stereoscopic viewing and follow, passively, what one active person is controlling by utilizing the VR devices.

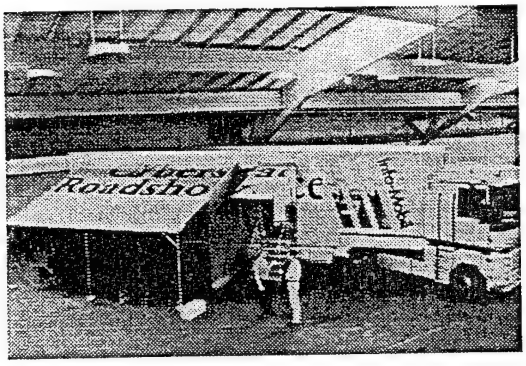


Fig. 1: Roadshow truck with VR equipment

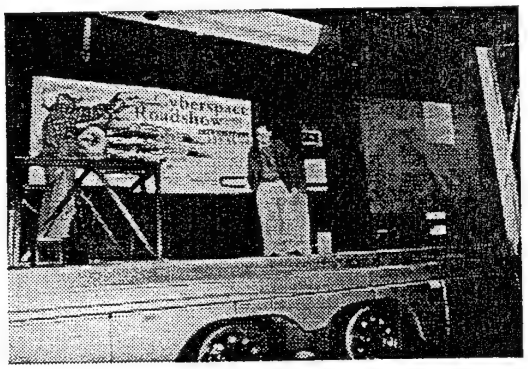


Fig. 2: Presentation stage with action point (left) and stereoscopic rear projection (right)

The Multi-Channel Option delivers a high-resolution video signal (1280 x 1024 pixel) to drive the stereoscopic projection in highest quality. With a scan converter one signal is transformed into a standard low resolution NTSC signal to feed the HMD. To reduce the image generation complexity we decided to run the HMD in monoscopic mode, because former presentations prooved that due to the poor resolution of the HMD used, the appealing thing is its wide field of view and not the stereoscopic viewing. This will certainly change when HMD's with higher resolution are available/affordable.

Being aware of the fragile VR peripherals in use, a replacment for the HMD and the tracking system was always on board. Keeping our experience with VPL's fibre optic glove for public presentations in mind, it was used purly as a platform to mount the hand tracker (Felger 1992). No gesture interfacing was applied and it was expected that the glove would not survive the entire event. It was really only included because it was the central component in the nation wide advertizing campaign.

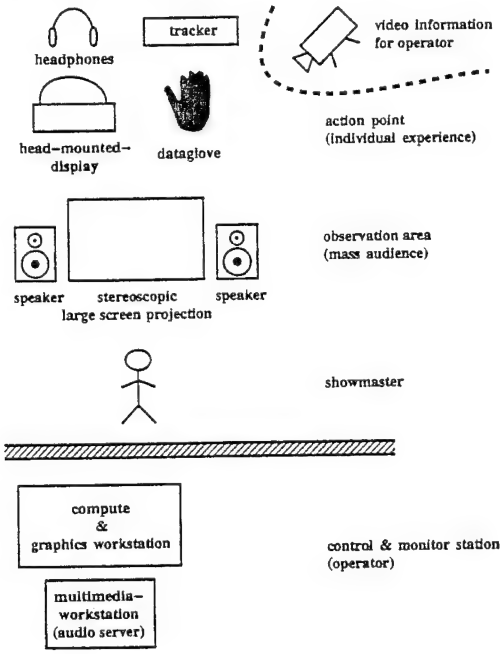


Fig. 3: VR infrastructure

Interaction With The Virtual World

Various actions should be available for the players when they experience a virtual world. This considers interactions with objects representing the virtual world as well as navigation within the world.

Static worlds, observed in typical walk-through applications, can get boring very soon. To create a vivid environment, dynamic object behavior has to be integrated in a virtual world. We introduced animations for lights, camera and any world object. Although these animations have to be precalculated (mostly) and thus the objects

do not react in an intelligent manner to the players' actions this makes the worlds much more interesting. Animations can be triggered or combined with user interactions or run continuously for as long as the player is within the virtual world. With these features the resulting presentation becomes a mixture of preprocessed animation and high-interactive virtual reality, both running in harmony under realtime requirements.

Navigating in virtual worlds requires in this special setup some new approaches. As the DataGlove had to be integrated any way possible because it symbolizes VR technology for the lay audience, we have to work around its inherent deficiencies. The DataGlove may cause problems because it has to be calibrated to each player's hand, and the applied fiber optic technology is not robust enough for permanent public use. Hygiene still is another issue. Another lesson learned from previous events and also strongly recommended by Prof. Henry Fuchs from UNC during his talk at IGD in April 1993 is: Don't fly !!!

People have problems with standard navigation mechanisms based on the DataGlove (point & fly). Thus we realized a carpet metaphor for navigation, where the players can move freely with respect to the constraints of cable length and tracker range (i.e., physical borders of the action point limit the movements) to perform small-scale navigation. To travel longer distances they are transported or beamed in a predefined manner. This is similar to the Aladdin realization at EPCOT done by Disney Imagineering except that we do not use a real carpet and do not have the capability to control the carpet.

The players trigger the movement over long distances in general by touching an appropriate object in the virtual world. This object represents the movement behavior (e.g., moving along a specific path for a certain time). Moreover, such a basic interaction mechanism is generic and can be applied and configured for different virtual worlds (no explicit programming necessary). This is realized as a list of touchable (triggerable) objects with associated actions. Initial actions can be started when entering or switching to a new virtual world. The following actions can be distinguished (combinations are possible too):

1) Animation of objects and lights in terms of transformations. Animations can loop endlessly or according to a given repetition counter.
2) Animation of the camera (view of the observer) in terms of transformations. Various modes are possible:
   - Additional head movement is allowed: yes or no.
   - Interpolation from a current camera view to a given starting view is enabled: yes or no.
   - Accelerate or slow down the camera movement.
   - Camera animations can loop endlessly or according to a given repetition counter.
3) Trigger a sound event.
4) Switch between two virtual worlds.
5) Switch the visibility of objects between visible and invisible.
6) Animate an object in terms of changing its geometric representation. An alternating object sequence can loop endlessly or according to a given repetition counter.

Presentation Story

The idea of the story was to generate an eventful and complex adventure world consisting of several subworlds which were connected by a simple transition structure. Furthermore, with respect to the expected players and audience, the Junior Bank Card was to play a major role in the story. With the integration of special effects (see below) a certain atmosphere or mood is created for each world. Due to time pressure, only a few ideas could be realized (e.g., virtual rain was skipped). In general, sound helps a lot to create a mood and is used throughout the adventure world (Astheimer 1993).

At the beginning, the players were flying, helicopter-like around the truck (see Fig. 4) until they were transported towards the action point inside the truck. This symbolizes the shift from the real world into the virtual real world and represents the start of the virtual journey.

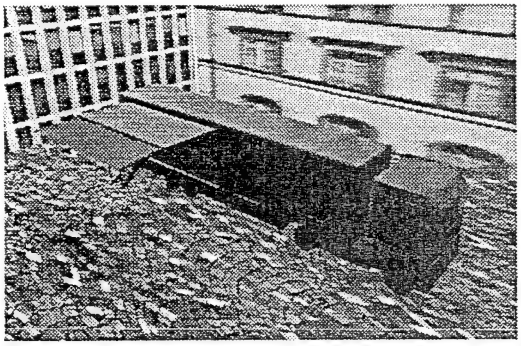


Fig. 4: Helicopter flight around the virtual truck

After arriving at the action point the players faced a contomat (SBG's teller machine, see Fig. 5). Up to this point they were completely guided by the system, which now changes to interactive mode. By touching the contomat with the hand the players triggered the transition from the virtual real world (truck with environment) to the virtual imaginative worlds.

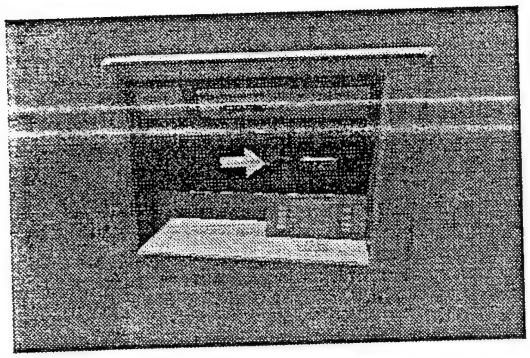


Fig. 5: Contomat (teller machine)

The players were then sucked into the card feeder and found themselves flying through a tunnel (see Fig. 6). For a certain time they proceeded through the tunnel with good (blue sky texture) and bad (yellow/red texture) objects (cubes) approaching them. They should have tried to touch the good objects and avoid contact with the bad ones. Appropriate sound signals were generated in both cases. The bank card was always flying ahead, guiding the players through the tunnel and releasing particles, but never reachable.

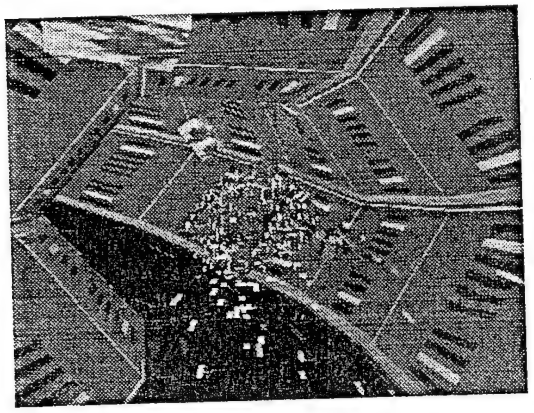


Fig. 6: Tunnel flight

Next, the players approached a switch room (see Fig. 7) where they selected one out of three alternative worlds in order to proceed with the journey. The bank card dissolved into three parts which disappeared into these three worlds. The choice between the corresponding virtual worlds was symbolized by color and a typical object from each world (a palm tree indicated the jungle world, trumpets indicated the stage world, a planet indicated the space world). The players made their selection to enter one world simply by touching one of the big arrows just beneath them. We called this switch room the "tower of fate" because its design was inspired by the star-wars scene where the Jedi master fought Darth Vader on a narrow bridge overlooking a precipice. Here, spaceship-like flashing control lights were displayed.

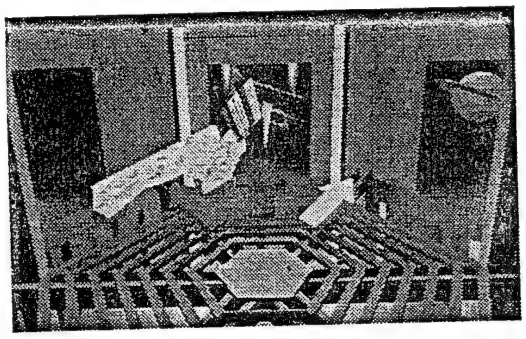


Fig. 7: Switch room (tower of fate)

In the jungle world (see Fig. 8) an elephant ride through a jungle was simulated. The players could touch animals which lived in this jungle. By doing so they were collecting parts of the bank card, whose fragments were attached to the back of the player's hand. Each successful hit was signaled by a sound event.

Fig. 8: Jungle

The stage world (see Fig. 9) provided a scenario where the players flew across a fictitious audience onto a stage, where they were surrounded by fantastic instruments (e.g., keyboard, guitar, chorus). Furthermore, a light show was performed on the stage. By touching the instruments, a short melody fragment from a song was played and the instruments moved as long as the music played. Keeping the instruments randomly playing rewarded the players with parts of the bank card. (Our original idea was to force the players to remember and replay a given melody sequence. But this was considered too restrictive to be solved by everyone.)

The space world was placed in a universe with moving asteroids and planets. It was designed as a labyrinth with five nodes and multiple connections (e.g., tube, staircase) between these nodes (see Fig. 10). Each node consisted of four exits. Touching one exit meant selecting this connection to the neighbor node and being moved along its path. In some nodes parts of the bank card were placed, where they could be collected.

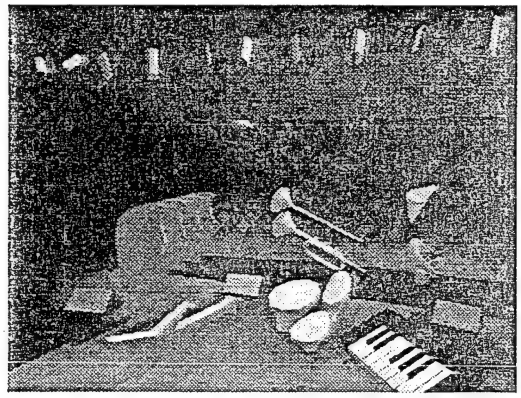


Fig. 9: Stage

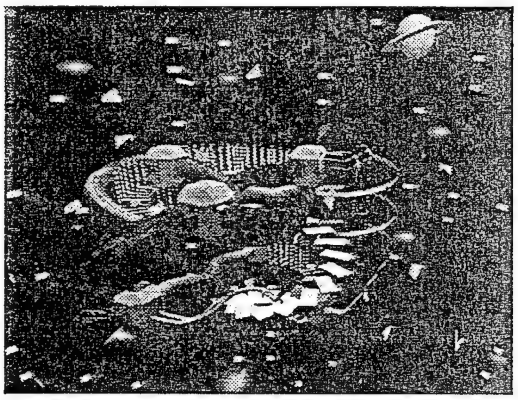


Fig. 10: Space labyrinth

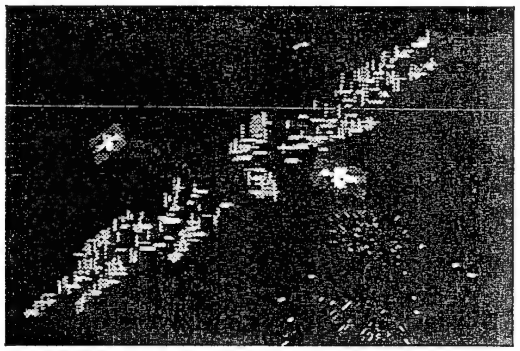


Fig. 11: Firework

Finally, after a certain time the entire experience ended with a colorful and loud fireworks display (see Fig. 11).

EXECUTION PHASE

Show Procedure

The roadshow toured across Switzerland from May 16 until June 15, 1994, with stops in Zurich (May 16-18), Winterthur (May 19), Aarau (May 20-21), Geneva (May 24-26), Lausanne (May 27-29), Sierre (May 30), Bern (May 31 - June 2), Zug (June 3-5), St. Gallen (June 6-7), Chur (June 8-9), Lugano (June 10-12), and Lucern (June 13-15). In each city a local group of volunteers from the SBG supported the event. At some locations the show was held inside a festival or exhibition hall, at others directly in front of the SBG building or in a public place. The entrance regulation was at the discretion of the local group. Some charged bank card holders 5 SFR and non-holders 15 SFR, others gave card holders free entrance. Mostly one free drink was included.

A typical presentation day featured 50 minute shows with a 10 minute break between two shows, running from 4 pm until 10 pm. Mostly, IGD staff gave a very brief introduction to VR technology and its applications at each show start (Encarnacao et al. 1994). A show master was guiding the players and the audience with respect to multi-lingual Switzerland, either in French, German, or Italian. Performing one VR experience together with interviewing the player took approximately 5 minutes time. Most of the time an audience of between 10 and 50 persons attended one show. We had days with a total of roughly 400 persons sharing the presentations. Players were selected by the show master or by means of a lottery. The players started with the sequence of the virtual truck, and the approach to the contomat; this was followed by the tunnel flight, the switch room, and either the space labyrinth, the stage, or the jungle ride; it concluded with the fireworks. Each player was rewarded with a unique "Cyberspace Roadshow T-Shirt".

Media Coverage

The roadshow was promoted by a nationwide advertizing campaign. This included posters, radio spots by local radio stations, and a preview in SBG's newsletter "megascene". Furthermore, two press conferences were held, one at the tour start in Zurich, and the other one later in Geneva.

The response in the media was enormous and very positive. Already, during the first week the roadshow had coverage in most major Swiss newspapers and the public TV station. This continued steadily over the full runtime of the

event. In total, approximately 100 press articles can be counted, as well as four TV spots (even one from an Austrian TV station), and five radio interviews.

Observations

After watching the large-screen stereoscopic projection in highest resolution, many players were disappointed about the poor resolution of the HMD (90k primary pixels). Some complained about the integrated system guidance or expressed the feeling that they were not really mastering the world (sometimes the operator "helped" to interact via keyboard function keys). Another complaint was about the jaggy motion due to low frame rates (approximately 5-10 frames/sec. depending on the world complexity) and tracker faults, which resulted in annoying pauses.

As in numerous earlier presentations, it was very hard to explain that all computations and computer graphics were done online under real-time requirements and no prerecorded tape or video is involved. On the other hand many people were very interested in this, starting some PC-level discussions to figure out the enormous computational work load.

We got all types of persons (e.g., from kids in kindergarten age to retired adults, male/female, computer-educated/non-computer-educated).
Most people did like the experience, a few were enthusiastic, a few did not like it at all. Groups cheered on the players. Some people were very passive and showed a TV-like stiff behavior, some were very active (too active in fact for the tracking system). Very tall or small people (kids) did not match our predefined settings and they sometimes had extraordinary perspectives. The most popular world was the space labyrinth, where surfing the staircase up or down was especially appealing.

Lessons Learned

A roadshow and its show locations have to be planned very carefully. Luckily, we did not have to face unsolvable problems but it seems to be wise to check the locations well in advance by the truck driver. Although the enormous size of the truck was known to everybody, sometimes its delicate maneuverability was not considered adequate.

If you want many people coming to see the show, it is very important to place it somewhere in a central open air place, rather than at a remote location. For example, in Geneva the show was held at the fairgrounds (Palexpo) at the city periphery and was rather sparsely attended. At Lugano the truck stopped in the main square near the lake front (Piazza Riforma) and it was almost impossible to handle the crowds waiting in the surrounding street cafes.

You should expect a great variety of people joining the show. Even when the target audience seems to be of a certain age (Junior Bank Card holder's age is less than 20), be prepared for everybody. Make a comprehensive dry run of your software and the created worlds with such a large variety of people, not only the ones available in your lab. If possible, be flexible enough to adjust your software or world during the show.

You should set aside enough time for the design and creation of your virtual worlds. Spend at least half of the development effort on this and you will get creative worlds. Less than three month for the preparation and realization of such a project was extremly short and could only be tackled by a highly integrated, cooperative team.

Our biggest surprise was not having any major problems with the VR infrastructure. The complete equipment ran without failure during the roadshow. There was not one show which had to be delayed or cancelled. But it will be reassuring to have redundant equipment with you. Running the hardware over months and years in our lab tells us that you can not expect such a nice behavior.

Although HMD and DataGlove are the classical VR symbols, think twice about whether the intended presentation will really need it. For this roadshow it was a requirement and we dealed with it by implementing dedicated interaction techniques which worked fine. If you have the freedom to choose the equipment, wait for really robust and comfortable HMD's and gloves, or avoid their use.

FUTURE WORK

During the roadshow and the weeks after the roadshow IGD received many inquiries about this Cyberspace Roadshow. Many had the same goal, to use a mobile VR system for marketing purposes. With a few companies which can afford such an installation, further negotiations are ongoing. You can expect one future event in mid 1995 in Germany. Confidentiality reasons prohibit a disclosure at this time.

As IGD is a research institute, we are always looking for new challenges and do not want to duplicate finished projects. We are improving our VR system with respect to requirements for entertainment and leisure applications. The integration of a personal motion platform lies within a medium-term time scale.

CONCLUSIONS

This paper described the Cyberspace Roadshow which was performed for the Schweizerische Bankgesellschaft (SBG) in Switzerland. The associated work in preparing, realizing, and executing this roadshow were described in detail as a case study.

For this project the interaction concept of IGD's VR system was redesigned and extented. Interactions can be specified in a data file referencing world objects and predefined actions. Thus interesting living or reacting worlds can be easily defined and evaluated. It has been prooved that a mobile, high-quality, immersive VR installation is feasable. Moreover, recently a mobile driving simulator has been prototyped (Latham 1994).

For the SBG this marketing event was a big success, although the response was not as big as predicted. As usual for marketing events, SBG did a questionaire along with the roadshow, but the results are not available to the authors. According to the SBG, especially the extensive media coverage accross Switzerland has strenghten significantly SBG's image of dealing with innovations and being known for new ideas "off the beaten track". SBG spent about 250,000 SFR to enable this event (NN 1994).

ACKNOWLEDGEMENTS

We acknowledge our very ambitious and hard-working team (in alphabetical order): Peter Frisch, Torsten Fröhlich, Wolfram Kresse, Rolf Kruse, Stefan Müller, Uli Spierling, and Johannes Strassner. We are grateful to Werner Vieth for his permanent care of the projection system and to the truck driver Hans Scheidt for doing such a good job and sometimes even making impossible manouvers possible. We would like to thank the show masters Nicole Calame and "Rookie" D. Ruckstuhl for their great help. Our gratitude belongs to Stefan Merz from the agency Bosch & Butz, as well as to Simon Stauber and D. Zoppi from the SBG for having the roadshow idea, supporting it, and their believe in its success. Special thanks to Mike Sokolewicz for proofreading. He is responsible for much of the readability and none of the faults.

REFERENCES

Astheimer, P. 1993. "What you see is what you hear - Acoustics applied to Virtual Worlds", In Proc. of IEEE Symposium on Research Frontiers in Virtual Reality (San Jose, USA, October 1993) pp. 100 - 107

Astheimer, P., Felger, W., Müller, S. 1993. "Virtual Design - A Generic VR System for Industrial Applications", Computers & Graphics, Pergamon Press, vol. 17, no. 6, November 1993

Blinn, F.F. 1994. Animation Tricks, Siggraph course notes no. 1, 1994

Cohen, M. 1994. "Cybertokyo: A survey of public VRtractions", Presence 2(4), 1994

Dodsworth, C. et al. 1994. Digital Illusion: Distributed Interactive Entertainment, Siggraph course notes no. 7, 1994

Encarnacao, J.L.; Astheimer, P.; Dai, F.; Felger, W.; Göbel, M.; Haase, H.; Müller, S.; Ziegler, R. 1994. "Virtual Reality Technology - Enabling New Dimensions in Computer-Supported Applications", In Proc. of 9th Japan-Germany Forum on Information Technology (Oita, Japan, Nov. 8-11) 1994

Felger, W. 1992. "How interactive visualization can benefit from multidimensional input devices", In Proc. of SPIE/IS&T Symposium on Electronic Imaging Science and Technology, Conference 1668: Visual Data Interpretation (San Jose, USA, February 9-14) 1992

Felger, W.; Waehlert, A. 1995. "Employment Potential and Application of Virtual Reality in the Domain of Business Management", In Proc. of the Virtual Reality World '95, (Stuttgart, Feb. 21-23) 1995 (in press)

Giles, W., Schroeder, R., Cleal, B. 1994. "Virtual Reality and the Future of Interactive Games" (VR '94, Stuttgart)

Helman, J. et al. 1994. Designing Real-Time Graphics for Entertainment, Siggraph '94 course notes no. 14, 1994

Latham, R. 1994. "Illusion Technologies REAL DRIVE Simulator", Real Time Graphics, Vol. 3, No. 4, 1994, pp. 1,3,14

N.N. 1994. "Der Kunde im Wunderland", acquisa, August 1994, pp. 32-34 (in German)

VIRTUAL ENVIRONMENT TECHNIQUES FOR THE EXPLORATION OF EARTH OBSERVATION DATA

Frank Bagiana
Simulation Facilities Section
ESA/ESTEC
P. O. Box 299
2200 AG Noordwijk, The Netherlands

Hans Jense
High performance Computing Department
TNO Physics and Electronics Laboratory
P.O. Box 96864
2509 JG The Hague, The Netherlands

Abstract

Remote sensing (RS) satellites produce large amounts of scientific data. With the outlook of many more RS sensors to come and the massive amount of data related with that, the need for good visualisation tools is stronger than ever. Virtual Environments (VEs) are promising tools to perform visual analysis of this type of data in an intuitive manner. A prototype system is described, which employs VE technology, for the interactive visualisation of remote sensing data. The system facilitates the interactive visualisation of multiple large remote sensing datasets from different sources, as well as remote sensing datasets in relation with data originating from other sources, e.g., terrain data from geographic information systems. The system is based on an existing visualisation system (IRIS Explorer) that has been enhanced with several data management and rendering functions. Data management functions include region-of-interest selection, geometric data correlation, image data filtering, and geometry decimation. The existing Explorer rendering capabilities have been enhanced with motion parallax based on head-tracking, texture-mapping functions to combine image data with geometric terrain models, and dynamic adaptation of display resolution. These functions allow the user to display significantly larger amounts of remote sensing data at real-time frame rates, and interactively control the trade-off between real-time (rendering speed) and realism (data complexity).

1. INTRODUCTION

Researchers in the physical sciences nowadays have to deal with massive amounts of data originating from both experiments and simulations. The problem of exploring these data in search of meaningful results has recently been recognised. The National Science Foundation (NSF) report "Visualization in Scientific Computing" [1], explicitely mentions earth resource satellites as sources of incredible amounts of data. Obviously, the increasing number of satellites, both for earth observation and for exploration of other planets, as well as the increasing data density of these information sources will require the use of more advanced visualization techniques if scientists want to continue examining these data in a useful way. Current scientific visualization systems still rely on conventional, window based 2D user interfaces. The system described here aims at introducing user interface techniques from virtual environment (VE) technology in scientific visualization applications. The combination of interactive stereoscopic display with the ability to intuitively change the point-of-view, allows the user to move around his data sets more easily, freeing him to concentrate on the exploration of their contents. Our system is a prototype, designed to be open towards future extension into a more general system. It incorporates a number of important elements of virtual environment technology, most notabaly stereoscopic display, including head-slaved motion parallax. It allows the interactive exploration of large remote sensing data sets in an intuitive manner.

The remainder of this article is structured as follows. Section 2 provides some background on various sources of remote sensing data and current visualisation practices. The architecture of the prototype system is described in section 3. In section 4 some preliminary results are shown. Conclusions are drawn and suggestions for future work are made in section 5.

2. BACKGROUND

2.1 Remote sensing

In 1991 the European Space Agency (ESA) launched the ESA Remote Sensing satellite-1 (ERS-1). Since that time a huge amount of data has become available to the user community. ERS-1 has a number of sensors among which the Synthetic Aperture Radar (SAR), the wind scatterometer (WSC), the radio altimeter (RA), and the along track scanning radiometer (ATSR). These sensors produce data of a very different nature. Many geophysical parameters can be obtained from these sensors. To name a few: sea surface temperature (ATSR), sea surface winds (WSC, RA), mean sea surface (RA). In the near future, satellites like the ERS-2 and ENVISAT will add many new sensors to this list. This will dramatically increase the amount of information available about our planet. It is a challenging task for scientists to translate this information to products which can be used by end-users of Remote Sensing data. Comparing RS data with either data from other sources or with other RS data might provide new insights in their relations. From this knowledge new applications from RS data can be generated. Virtual reality is a very promising tool to perform this type of data fusion in an intuitive manner. Especially with the outlook of many more RS sensors to come and the massive amount of data related with that, the need for good visualization tools is stronger than ever.

2.2 Scientific visualisation

Over the last years visualization has become an accepted tool for scientific researchers. The motivation for using visualization techniques is the ever increasing deluge of data that researchers have to deal with. The size of these data sets makes it impossible to explore, analyze and understand their contents unless they are represented in pictorial form.

Two broad categories of visualization systems can be distinguished. The first consists of more-or-less "closed" application programs. An example from this category is the Data Visualizer from Wavefront. Open application construction toolkits form the second category. They allow the user to assemble a visualization application from predefined modules, each of which implements some basic function. By feeding the output from one module into another (or into several others), networks of function modules may be created that implement a particular visualization application. Visualization toolkits from the second category are AVS [2], and IRIS Explorer [3]. Recently, another interesting approach has been described by Hasler et. al. [4]. They have developed a spreadsheet-like tool specifically aimed at organising, browsing, and manipulating large nubers of RS images. Images are organised in a matrix of cells, and the user can interactively specify formulas to derive new images, using image processing, image analysis, and rendering primitives, analogous to the use of traditional spreadsheets to organise and manipulate text-based data.

When the user has facilities at his disposal for the manipulation of data sets and can influence the way in which they are depicted, with a response time between his actions and the display of the resulting image that is sufficiently short, he is effectively "put in the visualization loop". Because of the size of the data sets involved and the large amounts of computing resources required for this, systems for interactive visualization need to offer ways in which the user can select different trade-offs between image detail and response time. If such a system has been well designed it allows the user to interactively explore huge data sets, quickly focus on areas of interest that may require more detailed study and on the whole analyze his data much more effectively then by "brute force" numerical analysis methods.

3. THE SIEVE SYSTEM

Recently, TNO-FEL has been working on the development of a prototype system, employing Virtual Environment technology, for the interactive visualisation of remote sensing data. This project is carried out for ESA within the framework of its Technology Research Programme (TRP). The aim of the prototype is provide a "quick-look" tool for the remote sensing user community. For this system the name SIEVE (Scientific Interactive Exploration with Virtual Environments) has been proposed. The SIEVE prototype facilitates the interactive visualisation of multiple large remote sensing datasets from different sources, as well as remote sensing datasets in relation with data originating from other sources, e.g., terrain data from geographics information systems. In some sense, the SIEVE system is similar to the NASA Ames Virtual Planetary

Exploration (VPE) system, although this is specifically aimed at the visualisation of geometric planetary terrain models of, e.g., Mars [5].

3.1 System overview

The SIEVE prototype is based on IRIS Explorer, an interactive visualisation environment, running on Silicon Graphics workstations under the IRIX operating system [3]. IRIS Explorer is a modular interactive visualisation environment based on the data flow paradigm. All software components of the SIEVE prototype are implemented as Explorer modules.The linking structure between the modules is provided by the IRIS Explorer dataflow mechanism. This setup supports a modular and generic architecture with a high flexibility towards future adaptations or upgrades.

The prototype contains a set of functions that allow the user to visualise a number of data products from ERS-1, SPOT, and JERS-1 both from imaging and non-imaging sensors. All RS data can be combined with other RS data, or with digital elevation models (DEMs), e.g., from the Digital Land Mass System (DLMS) database, where appropriate. The user interface provides control over all data manipulation and rendering functions. The user can select (multiple) datasets to be visualised and controls the type of rendering operations. Attributes added to the data can be selected for visualisation to improve classification. The position of the observer in the virtual database world and his viewing direction are taken into account in the generation of stereoscopic images in order to provide a sense of "immersion" of the user in the RS data. Manipulation of the viewing parameters offers scaling and zooming in on details of the data. The user may interactively select the required level of detail of the visualised data set. Together these functions allow the user to concentrate on exploring the contents of the data in an intuitive way. A high-level functional blockdiagram of the system shows 3 main components (figure 1). The large amount of data invloved (a single ERS-1 SAR image consists of 8000 x 8000 16 bit pixels) and the requirements for real-time display, especially when head-slaved motion parallax is employed, demand appropriate data reduction facilities at each stage of the pipeline. The techniques employed are described in more detail below.

Figure 1. Main system components.

3.2 Reading

Figure 2 below illustrates the data reading facilities of the SIEVE prototype. In a given RS application, a user typically reads in a number of image data sets and a digital elevetation model (DEM) on which the image data should be mapped. In order to keep the amount of data read into memory managable, the user specifies a region-of-interest (in latitude/longitude coordinates) and a rate at which each image data set is sub-sampled. The Warp module does a geometric transformation and filtered interpolation to fit different images (possibly of different resolutions) onto each other (image registration). After having been read in, image data and terrain DEM's are internally represented as a standard Explorer data type called 'lattice'.

Figure 2. Data reading functions.

3.3 Mapping

The Mapping function translates the lattices resulting from the Reading function into other lattices or geometric primitives. For example: a SAR image is a 2-dimensional array of scalars, where the scalars represent the radar back scatter. The Mapping function allows the user to for instance map these scalars onto z-values so the radar back scatter shows up as height in an artificial DEM, or the user can map the scalars onto colour values. One particular mapping function, called DisplaceLat, is used to map the scalar height values of the digital elevation model to actual Z coordinates. The DisplaceLat module transforms a 2-D lattice with elevation data into a 3-D lattice, whereby the user can scale the elevation. The 3-D lattice containing geographical (lat, long, height) coordinates is then passed to the Projector module. The Projector module transforms the (lat, long, height) coordinates to a cartesian (X, Y, Z) coordinate system according to one of the map projections, that can be selected by the user.

The output of the Projector module is passed to a module, that transforms the lattice into a geometry representation. Explorer provides a standard module for this purpose, called LatToGeom, that uses a straightforward triangulation algorithm. This algorithm transforms an N x M element DEM into a triangle mesh of roughly 2 x N x M elements. We have enhanced the functionality of the LatToGeom module by incorporating a decimation algorithm. This algorithm groups together the coplanar triangles, that resulted from the triangulation process, to form larger triangles [6]. This reduction in the number of output triangles significantly speeds up the rendering process.

Figure 3. Data mapping functions.

3.4 Display

The interactive presentation of the geometry to the user is performed by the Display function. Given the geometry from the Mapping function and input from the user, like viewpoint, view direction and other viewing parameters, the Display function generates the image belonging to these inputs. The standard Explorer display module, called Render, provides facilities for, e,g, interactive manipulation of objects and lightsources, rendering parameters, etc.

When the standard Explorer functions are used the output of the mapping process is a "pure" geometry i.e. a description in terms of nothing but geometric primitives of the scene that will be displayed to user. Usually in remote sensing applications, when data sets from different sources are combined, they are of widely differing resolutions. For instance, an ERS-1 SAR image has a resolution of 12.5 x 12.5 meter, while a corresponding DEM of the same geographic area has a resolution of the order of 100 x 100 meter. When both data sets are read-in, transformed into lattices, and finally turned into geometric representations for rendering, it is the size of the high-resolution data set that determines the size of the resulting geometry, and thus the size of the terrain that can be displayed at reasonable frame-rates. When the hardware platform provides real-time texture mapping support, this can be exploited by mapping (part of) the high-resolution (SAR) image as a texture on the geometric representation of the digital elevation model. Although Explorer provides no standard functionality for texture mapping, the underlying Inventor software of the Explorer Render module does support it. We have extended the Explorer Render module interface with an input port that accepts a lattice which is to mapped onto the geometry that arrives at the standard geometry input port. The advantage of this approach is that the bottleneck of the rendering pipeline, which in this case turns out to be the geometry processing, is significantly widened, allowing the display of much larger areas from the remote sensing data sets than was previously possible.

A final enhancement to the Render module concerns the dynamic switching of display resolutions. The standard Explorer Render module provides a drawing mode called "move-lo-res". As long as the user moves the viewpoint, the geometry to be rendered is subsampled, resulting in much fewer triangles being rendered , and thus much faster frame rates. As soon as the viewpoint remains statis, the whole scene is rendered at full resolution. For our application, the implementation of this mode was considered unsatisfactory, because the input geometry is subsampled in only one direction. The result is that during movement, the terrain being rendered is shown as a collection of parallel strips. We have achieved resolution reduction, of the scene, by adding an extra input geometry port to the Render module, through which a uniformly subsampled version of the high-resolution geometry is input. Depending on whether the viewpoint is being moved or remains static, the Render module now chooses the appropriate input port (lo-res or hi-res). An additional benefit if this approach is that the resolution of the lo-res geometry is now under user control.

Figure 4. Display functions.

To provide a researcher with improved depth perception, he can (but does not have to) make use of the head-tracked stereoscopic display facilities.These are provided by the StereoGraphics CrystalEyes/VR product. The CrystalEyes stereo glasses enable the user to perceive depth by fusing two alternately displayed images, one for each eye.

Another depth cue can be obtained, when the user's head movements are tracked. When the head movements are incorporated in the viewpoint of the rendering module, head motion parallax of the rendered images can be

obtained. The sensor supported by the SIEVE system is the Logitech ultrasonic head tracker. This device is integrated in the CrystalEyes/VR glasses. The motion paralax gives an extra depth cue, that significantly enhances the depth perception. Both the stereo viewing and the head tracking can be used together to get optimal depth perception, but they can be used separately as well [7].

4. RESULTS

The target hardware for the SIEVE prototype is a Silicon Graphics Onyx workstation, equipped with the RealityEngine^2 graphics option. The target machine has two CPU's, and 128 Mbytes of main memory. The graphics subsystem comprises a single pipe with multi-channel option (MCO). The performance figures mentioned below were obtained on this system.

The use of the SIEVE system is best illustrated by means of an application. Typical example applications for the SIEVE prototype are:

1. Combined use of DEM and multi-temporal SAR data. A DLMS elevation data set is combined with multiple SAR images of the same area, but recorded at different times.
2. Combined use of SPOT and SAR: This application allows the user to visualy compare the contents of optical SPOT imagery with ERS-1 radar imagery.
3. Combined use of ERS-SAR and JERS-SAR. Combined use of multi-frequency, multi-temporal data is known to improve classification results.
4. Visualisation of multi-temporal change detection and optical imagery. In this demo two SAR images, taken a year apart, are used to detect changes in backscatter. The difference is displayed as height. This can then be compared with either one of the SAR images or with the SPOT image.
5. Visualisation of non-imaging sensors. Combined visualization of sea surface height, sea surface temperature, mean significant wind velocity. Correlations, if any, can be sought for.

For performance analysis purposes, two implementations of application 2 were made. In the first one, all image and terrain data were rendered as pure geometry, while in the second one, the images were texture mapped onto the geometric terrain model.

In the first trial te data sets all had a resolution of 200 x 300. The LatToGeom module produced a geometry of about 2 x 200 x 300 = 120K triangles. The Render module was able to render this geometry at a frame rate of approx. 4 Hz, with one head light and a Phong lighting model. This means we have obtained a performance of nearly 500K triangles per second, whereas the manufacturer claims 1.6M triangles per second, albeit unlit and flat-shaded. Thus, if we want to obtain a 25 Hz frame rate in this case, the geometry should be restricted to at most 20K triangles. In other words, the Digital Elevation Map should contain at most 10,000 elements. Taking the aspect ratio of a standard UTM grid (10 x 15 km) into account, this means the size of the DEM should be something like 80 x 120 pixels, where each element represents an area of 125 x 125 m., which is actually not too far of the original resolution of the DEM.

During a second trail, the texture mapping capabilities of the hardware were used. The AddTexture module enables the combination of DEMs with satellite images with different resolutions. As explained above, DEMs generally have a resolution of approx. 100 x 100 m, whereas ERS-1 SAR images, for instance, have a resolution of 12.5 x 12.5 m. Using a DEM with a size of 60 x 120 elements, resulting in over 14,000 triangles, yielded a frame rate of 33.3 Hz, independent of the texture size, which varied between 512 x 512 texels up to 1024 x 2048 texels. A DEM almost twice this size (80 x 160 elements) resulting in over 25,000 triangles, yielded a frame rate of 16.7 Hz. These results corroborate our previous performance estimates, based on the results of the first trial.

Stereo display was not used during the performance trials. Stereo viewing of course requires two images to be drawn, one for each eye. This means, that the frame rate for stereo projection will be halved.

Figure 5 shows an image of application 2, the combination of optical SPOT with ERS-1 radar data. The area covered by these datasets is in the vicinity of Zwolle in the Netherlands. The river shown in the center is the

IJssel. The hills in the left part of the image are the foothills of the Veluwe range. Note that terrain height is exaggerated. The use of the stereoscopic workstation for the interactive navigation of the terrain models is shown in figure 6.

Figure 5. Perspective rendering of optical SPOT and ERS-1 SAR images on a 3D terrain model

Figure 6. Stereoscopic workstation in use.

5. CONCLUSIONS AND FURTHER WORK

We have described a system for the interactive stereoscopic virusalisation of rempote sensing data. A number of data reduction methods are employed to meet the requirements for real-time rendering. All of these methods can be controlled by the user. This allows the interactive selection of an optimum trade-off between real-time (rendering speed) and realism (data complexity) in a given application.

The performance measurements described in section 4 were obtained whith a Phong lighting model. Later measurements on an IRIS Indigo gave a performance improvement with a factor of two when we switched to a lighting model using base colors. Although it is dangerous to assume the same performance gain on a different machine, some further performance improvement may be expected on the target hardware as well. Finally, the geometry decimation facilities were not used during the trials. This means that a further reduction of the number of triangles, and thus a performance increase can be expected.

Currently, SIEVE renders stereoscopic image pairs, which are presented to the user through liquid crystal shutter glasses. This method currently provides a good compromise between image quality and a sense of immersion through the stereoscopic depth illusion. Eventually we would like to be able to use a head-mounted display in order to achieve total immersion in the virtual environment. However, the requirements put by scientific visualisation applications with respect to resolution and field-of-view are currently beyond what is offered by available HMD's.

Extending the 3D presentation facilities of SIEVE with 3D manipulation capabilities is a next step which requires the use of 3D interaction devices. A natural candidate for this would be the Logitech 3D mouse device. This allows the user to point within the "virtual holographic display" presented in front of the monitor and select or manipulate objects. The addition of facilities for the automatic selection of level-of-detail and region of interest would further improve the capabilities of IRIS Explorer to deal with large data sets.

The SIEVE system is still under development. Therefore, no definite conclusions can be drawn at this time. However, because representatives from the remote sensing community are closely involved in the project, we feel confident that the result of the development effort will be a useful tool for them.

BIBLIOGRAPHY

[1] B. H. McCormick, T. A. DeFanti, and M. D. Brown, eds. "Visualization in Scientific Computing," Computer Graphics, Vol. 21, No. 6, Nov. 1987.

[2] C. Upson, et.al., "The Application Visualization System: A Computational Environment for Scientific Visualization," IEEE Computer Graphics and Applications, Vol. 9, No. 7, 1989.

[3] IRIS Explorer User's Guide, Document Number 007-1371-010, Silicon Graphics Computer Systems, Mountain View, CA., 1993.

[4] A. F. Hasler, K. Palaniappan, and M. Manyin, "A high performance Interactive Image SpreadSheet (IISS)," Computers In Physics, Vol. 8, No. 3, May/Jun. 1994.

[5] Micheal W. McGreevy, "Virtual Reality and Planetary Exploration," in: Alan Wexelblat, "Virtual Reality: Applications and Explora-tions," Academic Press, Cambridge, MA., 1993.

[6] W. J. Schroeder, J. A. Zarge and W. E. Lorenson, "Decimation of Triangle Meshes", ACM Computer Graphics, Vol. 26, Nr. 2, (Proc. Siggraph), July 1992.

[7] Micheal Deering, "High Resolution Virtual Reality", ACM Computer Graphics (Proc. SIGGRAPH '92), Vol. 26, No. 2, Jul. 1992.

ACKNOWLEDGEMENTS

The work described here was performed under ESA/ESTEC contract 10475/93/NL/JG(SC), and also sponsored by the Netherlands Agency for Aerospace Programs (NIVR) under contract NRT 2305 FE.

Further Development of the Responsive Workbench

Bernd Fröhlich Berthold Kirsch
Wolfgang Krüger Gerold Wesche

Dept. of Visualization and Media Systems Design
German National Research Center for Computer Science
Sankt Augustin, Germany
E-mail: bernd.froehlich@gmd.de

January 18, 1995

Abstract

The Responsive Workbench [8] is designed to support end users as scientists, engineers, physicians, and architects working on desks, workbenches, and tables with an adequate human-machine interface. Virtual objects are located on a real "workbench". The objects, displayed as computer generated stereoscopic images are projected onto the surface of a table. The participants operate within a non-immersive virtual environment. A "guide" uses the virtual environment while several observers can watch events by using shutter glasses. Depending on the application, various input and output modules have been integrated, such as motion, gesture and voice recognition systems which characterize the general trend away from the classical multimedia desktop interface.

The system is explained and evaluated in several applications: A virtual patient serves as an example for non-sequential medical training. The car industry benefits from areas like rapid prototyping for exterior design and interactive visualization and examination of flow field simulations (virtual windtunnel, mixing processes). Visualization and verification of experiments with mobile instrument deployment devices in outer space missions are another fascinating application. Architecture and landscape design are another discipline well suited for the workbench environment.

1 Motivation

The standard metaphor for human-computer interaction arose from the daily experience of a white-collar office worker. For the last 20 years desktop systems have been enhanced more and more, providing tools such as line and raster graphics, WIMP(Window Icon Mouse Pointer) graphical user interfaces and advanced multimedia extensions. With the advent of immersive virtual environments the user finally arrived in a 3D space. Walkthrough experiences, manipulation of virtual objects, and meetings with synthesized collaborators have been proposed as special human-computer interfaces for the scientific visualization process. Specific interfaces, originally developed for pilots and telepresence tasks, became available to the ordinary user (see [7], for example).

The dream of the ultimate medium, which uses all channels of human perception, has guided the efforts of user interface design towards these virtual reality systems. Unfortunately, head-mounted displays, body-tracking suits, and force-feedback exoskeletons are obstrusive. These systems separate the users from each other. Especially in scientific visualization applications, comprehensive attempts have been made to overcome these drawbacks. The BOOM systems allow for easy-to-use walkthrough and object manipulation experiences [3]. The surround-screen projection-based virtual environment CAVE [2] was designed for several users to become immersed with their whole body in a virtual space.

All these approaches to future user interface systems have one point in common: design of an (almost) universal interface based on the most advanced computer and display technology available.

Another approach to the design problem for future human-computer interfaces is rigorously centered on the users's point of view. Myron Krueger pioneered this attempt with his work on non-immersive responsive environments [7]. Application-oriented visualization environments have been proposed and built to support a specific problem-solving process. The computer acts as an intelligent server in the background providing necessary information across multi-sensory interaction channels (see [4], [10], for example).

We developed the Responsive Workbench concept, first described in [8], as an alternative model to the multimedia and virtual reality systems of the past decade. Analyzing the daily working situation of such different computer users as scientists, architects, pilots, physicians, and professional people in travel agencies and at ticket counters, we recognized that there is only small acceptance of a simulation of working worlds in a desktop environment. Generally, users want to focus on their tasks rather than on operating the computer. Future computer systems should use and adapt to the rich human living and working environments, becoming part of a responsive environment.

2 System description

During the analysis of the working environment and of the behaviour of the specialists, we recognized that the (cooperative) tasks of this class of users relies on a "workbench" scenario. The future impact of desk-like user interfaces in general has been discussed in [9]. Using a beamer, a large mirror and a special glass plate as table top, we built an appropriate virtual environment.

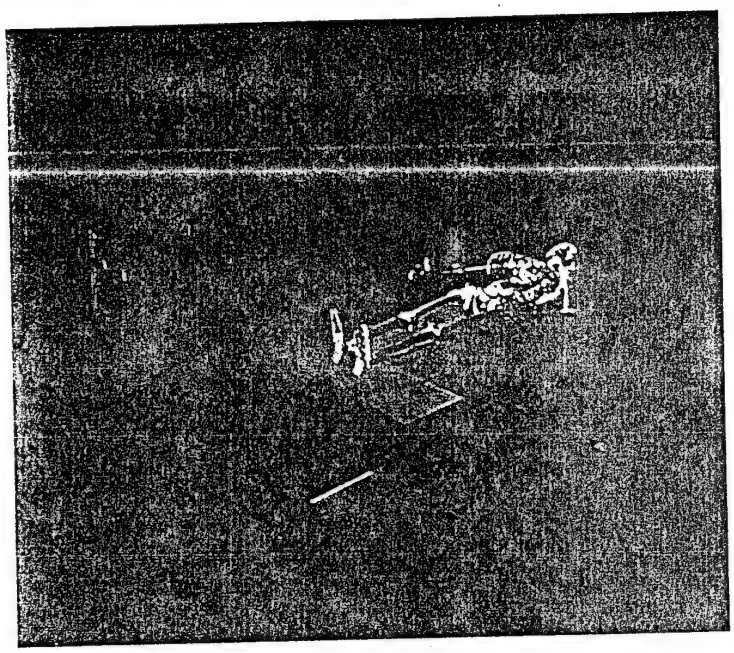


Figure 1: Set-up for a stereoscopic display of virtual objects on a desk

Virtual objects and control tools are located on a real "workbench" (see Figure 1). The objects, displayed as computer generated stereoscopic images, are projected onto the surface of the workbench. The projection parameters are tuned such that the virtual objects appear above the table. Depending on the application, various input and output modules can be integrated, such as motion, gesture and speech recognition systems. A responsive environment, consisting of powerful graphics workstations, tracking systems, cameras, projectors and microphones, replaces the traditional multimedia desktop workstation.

The most important and natural manipulation tool for virtual environments is the user's hand. Our environment depends on the real hand, not a computer-generated representation. The user wears a data glove with a Polhemus sensor mounted on the back. Gesture recognition and collision detection algorithms, based on glove and Polhemus data, compute the user's interaction with the virtual world objects.

To get correct stereoscopic rendering from any location around the workbench the system must keep track of the guide's eye positions. We realized this by mounting a Polhemus sensor on the side of the shutter glasses. It delivers position and orientation data for the head, allowing the system to calculate the position of each eye. Additional collaborateurs see the stereoscopic images with only slight distortions as long as they stay close to the guide.

The Responsive Workbench setup generates a very effective 3D impression which is due to the negative parallax, the wide angle of view and the head tracking. None of the users suffered from motion sickness using the workbench which happens often with head mounted displays. This seems due to the non-immersive nature of our approach. People

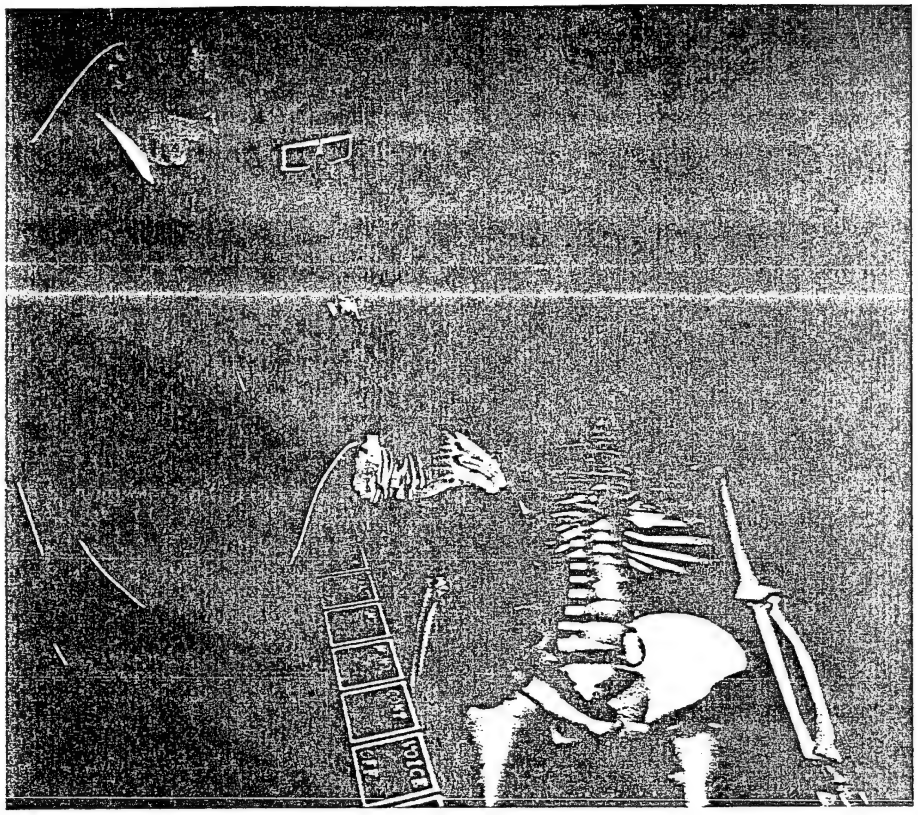


Figure 2: Cooperative work of a physician and a student

still have fix points in their environment so their senses don't get irritated.

3 Applications

Based on current research projects in the field of computer graphics, human computer interfaces and visualization, the following applications have been embedded in this new type of environment following the suggestions of the involved end users.

3.1 Medicine

3.1.1 Nonsequential training

This scenario is based on a real sized model of a patient. Figure 2 shows the model in a teacher/student scenario. The patient's skin can become transparent, making the arrangement of the bones visible. Now the surgeon or student can pick up a bone with the data glove and examine its joints, or take a closer look at the bone itself. The virtual patient could be examined in any detail through the zoom operation. Covered parts could be set free by removing the obscuring bones or organs with the hand or by making them transparent. Especially important for the understanding of many processes inside the human body are their dynamic aspects. We implemented two primary cases: the spatially exact reconstruction of the beating heart and the blood flow inside the transparent heart.

3.1.2 Simulation system for ultrasound heart examinations

This research project has been developed in close cooperation with the Center for Pediatry of the University of Bonn, Department for Cardiology, Germany. A typical user team is made from a radiologist, a surgeon and a visualization specialist.

Originally, the project was designed on a multimedia workstation. Recently we started to implement the system on the Responsive Workbench to meet the requirements of the surgeons for a virtual environment. They want to see the organ of interest and the measurement process in real or magnified size from all points of view in 3D space. They also would like to compare the simulation with the images on TV screens originating from the scanning process.

Detailed visualizations of the beating heart can be explored as interactive animations. The user can rotate the model in order to examine the structural and dynamic features of the heart. Different visualization modes (i.e., transparent, with/without blood circulation) are available. The complex interior structures and dynamics of the heart, valves, and blood can thus be examined (see Figure 3).

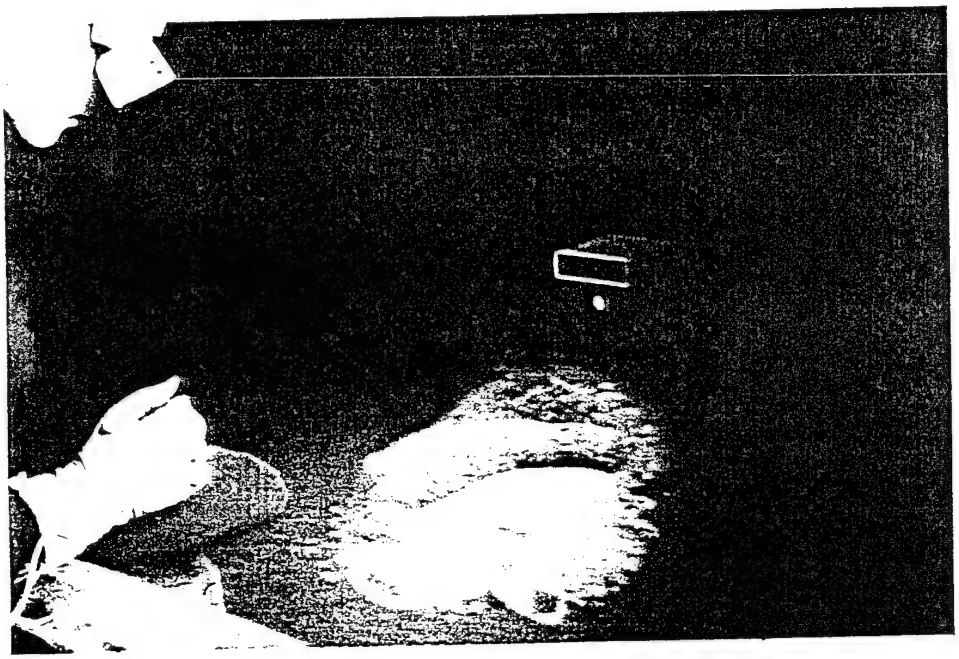


Figure 3: Examination of the blood flow in a human heart

3.2 Architecture and design

For the design and discussion process in architecture, landscape and environmental planning we implemented a basic testbed for demonstrations.

An architectural model is shown on the workbench, in our case the area around the buildings of our research institute. In front of the table two architects discuss the model,

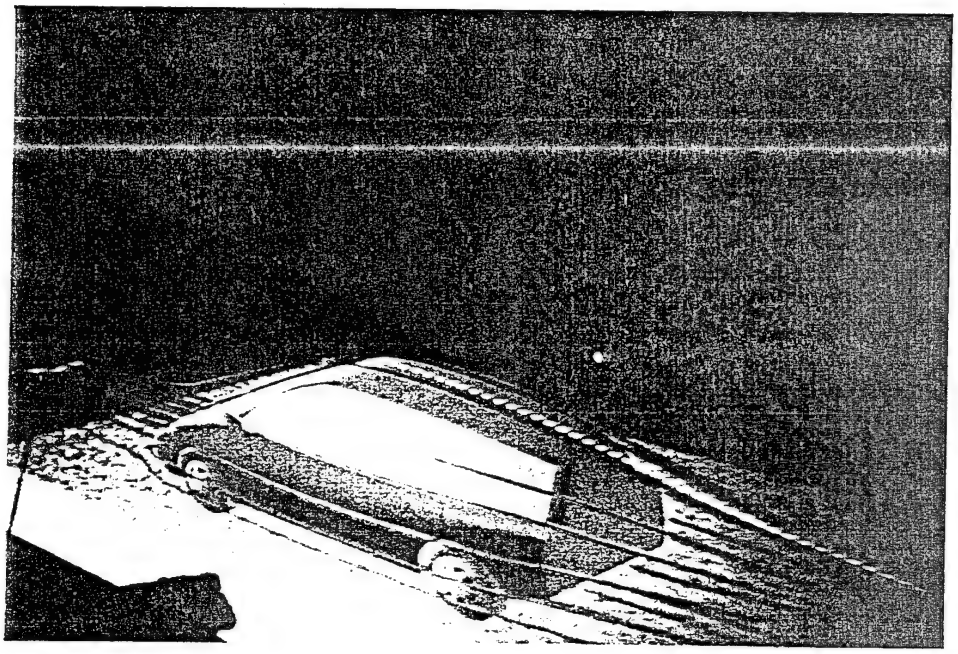


Figure 4: Virtual windtunnel scenario for car manufacturing applications (aerodynamical study model ASMO-II).

moving around buildings or other objects, such as trees in the virtual world. Additionally, lightsources can be set by the data glove to simulate different times of the day.

3.3 Automotive industry

In cooperation with scientists and engineers of the research department of Daimler-Benz AG, Stuttgart, we implemented two applications concerned with fluid dynamic simulations on supercomputers.

3.3.1 Virtual windtunnel

This application realizes the virtual windtunnel scenario [1] (see Figure 4) in the Responsive Workbench setting. The simulation data is taken from a finite element program running on a supercomputer or a highend workstation. In a preprocess the data points from the finite element mesh are resampled to a regular grid to speed up particle tracing. Particle tracing directly on finite element meshes is more accurate, but the additional computational cost restrict the number of particles, which could be handled simultaneously. The geometry data is also extracted from the finite element mesh and somewhat polished by a modeling system, e.g. by adding textures. A few precomputed streamlines are added as an overview of the flow field.

The stylus serves as a particle injector to examine any area around the car in detail. The particle generation rate and their lifetimes are adjustable. The velocity values of the flowfield are globally scalable even if this is physically not realistic.

3.3.2 Mixing process

The dynamics of the mixing process, generated by a supercomputer simulation, are visualized with the aid of fluid particles as rendering primitives. The essential physical properties to be visualized are the velocity field, pressure, temperature and fuel distribution. The mixing process is strongly time-dependent, so the data rate is much higher. The visualization shows the particle flow with color coded temperature during the injection process. These particle paths are precomputed during the finite volume simulation. The current implementation focusses on the interactive real-time exploration of the temperature and pressure distribution inside the cylinder with arbitrary cutting planes. The cutting plane is attached to the stylus which allows easy positioning. The finite element data is again converted to a regular grid, which serves as a 3D texture on the SGI Reality Engine 2 rendering system.

3.4 Simulation and control of outer space experiments

In cooperation with Deutsche Forschungsanstalt für Luft- und Raumfahrt e.V. (DLR) and other partners a mobile instrument deployment device prototype (IDD) will be developed.

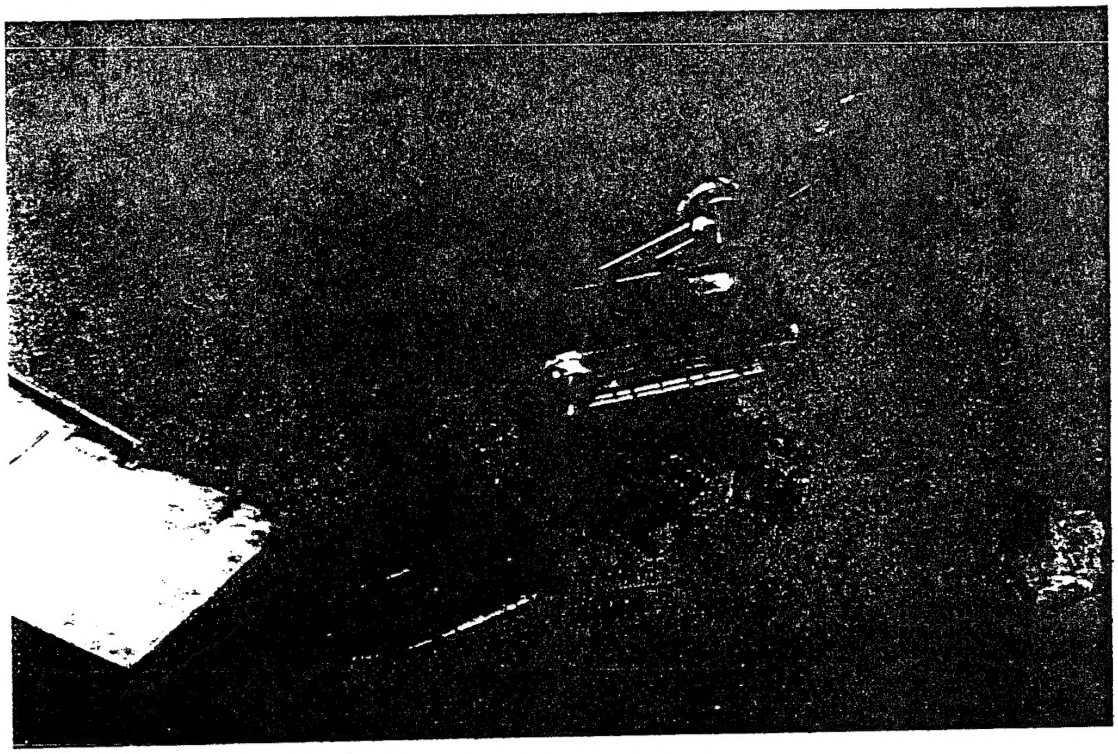


Figure 5: IDD TEM1 implemented by DLR and Uni Duisburg

A mobile IDD is a small microroboter for positioning of instruments on Mars or other space bodies to explore the near vicinity of the landing location. It is not possible to test the IDD under martian conditions or to control it on Mars directly. The first project stage studies the possible walking styles of an IDD and identifies the necessary data for a precise simulation of its behaviour. In a later stage the Responsive Workbench is meant

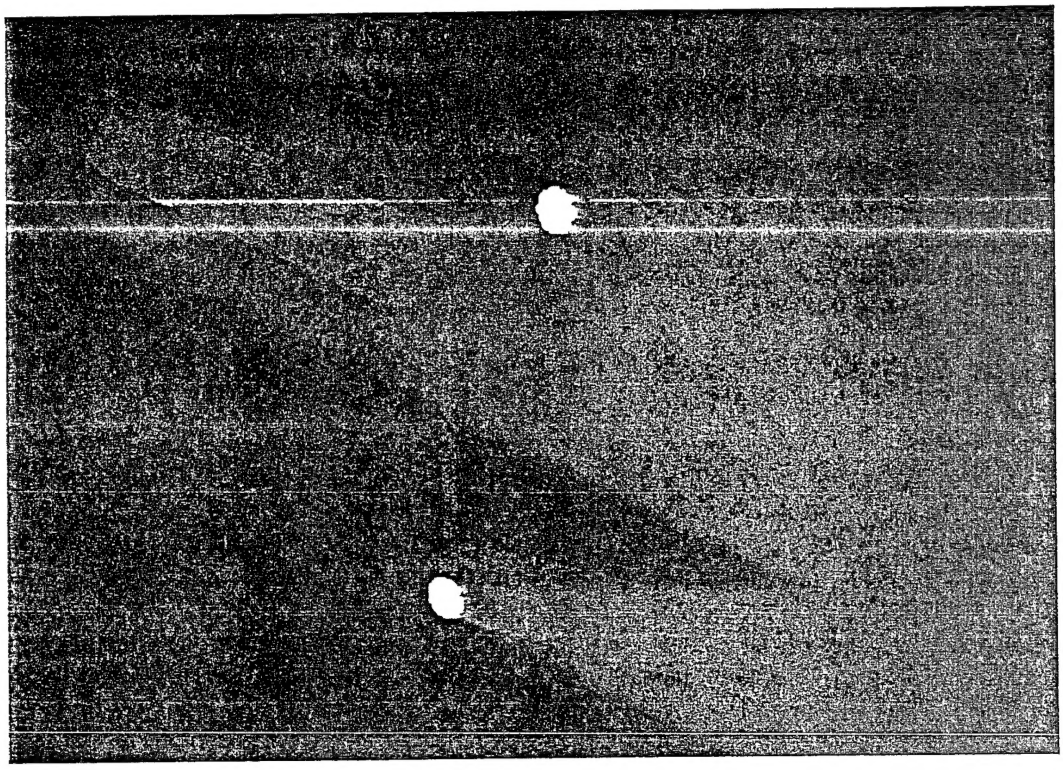


Figure 6: The computer model of the IDD

to display remotely sensed terrain data including the position of the IDD and the lander for simulation and planning of experiments.

An IDD prototype vehicle has been developed by Transmash, St. Petersburg. It consists of three container segments which are coupled by two traverses. In its smallest position the size of the IDD is about 35x20x7 cm. The IDD moves by rotation of the container segments which hold the instruments. The IDD has been further developed by DLR and the University of Duisburg (see Figure 5). Dynamics and kinematics are simulated using "MOBILE" [5], a multibody modeling system.

A computer controlled crawling or walking style can be developed in the Responsive Workbench environment. The main problems are: which moving styles are possible, which information (input sensors) is needed to control speed, direction and walking style or to program autonomous movement (reaction on obstacles, keep a given direction etc.) of the IDD robot [6].

Following the successful simulation of a save walking (crawling) path in the virtual environment at the ground station, the driving code is sent to the IDD operating on an other planet. Data measured by the IDD and the lander will e sent back to calculate the next steps and to update the visualization. This control loop is necessary to syncronize the remote and the virtual environment.

Typical operating sequences for an IDD are the approach to a preselected site, appropriate positioning of the instruments at the object, preparation of the object for measurements, measurement procedure, aquisation of a surface sample and analysis, digging to acquire a sub-surface sample and analyse it, return material to the lander for further

analysis, provide additional information for the selection of the next site.

The operating sequences are prepared and tested in the virtual environment. The lander station sends its data to the ground station. These data is used to construct the actual virtual world where the scientist acts. The scientist decides on the next action, teaches the new goal by i.e. pointing to the target site and runs the experiment within the virtual world. If the experiment has been successful the appropriate commands are sent to IDD on the planet. When the new situation on the planet has been incorporated into the virtual world the next sequence can start.

4 Conclusions

The Responsive Workbench system is designed to demonstrate the ideas and power of future cooperative responsive environments. Further applications under consideration running on this virtual workbench will be the simulation of air and ground traffic on airports, a training environment for complicated mechanical tasks, e.g., taking apart a machine for repair, landscape design and environmental studies via terrain modeling, and physically based modeling of virtual objects ("virtual clay"). These applications also rely on the workbench metaphor, but require specific interaction and I/O tools.

5 Acknowledgments

This work relies on the discussion with scientists and engineers of the research department of Daimler-Benz AG, Stuttgart, and physicians of the Centre for Pediatry of the University of Bonn. Especially, we are grateful to Prof. Redel for his involvement in this new field.

We thank our colleagues and students Stefan Banse, Manfred Berndtgen, Thorsten Fox, Klaus-Jürgen Quast, Peter Rohleder, Thomas Sikora, Josef Speier, Wolfgang Strauss, Jürgen Wind and Jürgen Ziehm for their extraordinary involvement in SW/HW management and modeling.

This work is partly supported by the Department of Research and Technology (BMFT) and the European Space Research and Technology Centre (ESTEC).

References

[1] S. Bryson, C. Levit, "The Virtual Windtunnel", *IEEE CG&A '93*, July 1992, 128-137.

[2] T.A. De Fanti, D.J. Saudi, C. Cruz-Neira, "A Room with a View", *IEEE Spectrum*, Oct. 1993, 30-33.

[3] I.E. Dowall, M. Bolas, S. Pieper, Fisher, J. Humphries, "Implementation and Integration of a Counterbalanced CRT-based Stereoscopic Display for Interactive Viewpoint Control in Virtual Environment Applications", *Proc. SPIE* 1256.

[4] M. Green, R. Jacob, "SIGGRAPH '90 Workshop Report: Software Architectures and Metaphors for Non-WIMP User Interfaces", *Computer Graphics*, July 1991, 229-235.

[5] A. Kecskemethy, "MOBILE User's Guide and Reference Manual", Fachgebiet Mechatronik, Universität Duisburg, 1993.

[6] B. Kirsch, U. Schnepf, I. Wachsmuth, "RoboVis - a Scenario for Using Virtual Reality Techniques in Learning Robot Development", *VISWIZ Report 4*, GMD, 1993.

[7] M. Krueger, "Artificial Reality II" *Addison-Wesley*, Reading, Massachusetts, 1991.

[8] W. Krüger, B. Fröhlich, "The Responsive Workbench", *IEEE Computer Graphics and Applications*, May 1994, 12-15.

[9] W. Newman, P. Wellner, "A Desk Supported Computer-based Interacting with Paper Documents", *Proc. of ACM SIGCHI*, May 1992, 587-592.

[10] J. Nielson, "Noncommand User Interfaces", *Communications of the ACM* 36, No. 4, (April 1993), 82-99.